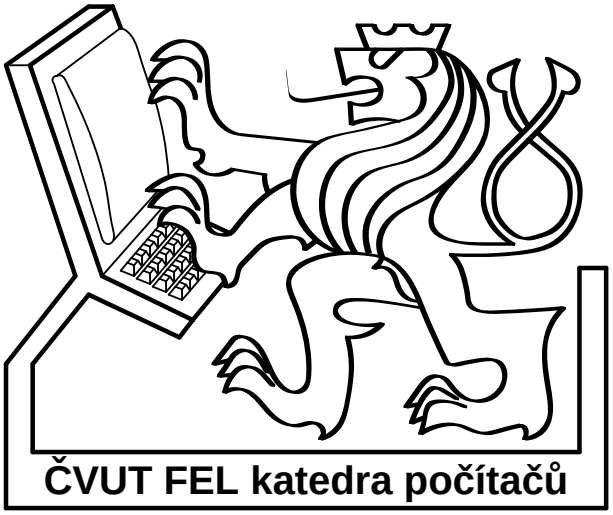
České vysoké učení technické v Praze

Fakulta elektrotechnická



Diplomová práce

Softwarové vybavení laboratoře pro světlotechnická měření

Tomáš Dvořák

Vedoucí práce: Dr. Ing. Libor Veselý

Studijní program: Elektrotechnika a informatika, dobíhající

Obor: Informatika a výpočetní technika

leden 2007

Poděkování
Děkuji svým rodičům za jejich obětavou podporu během mého dlouhého studia. Také děkuji Liborovi Veselému za cenné pracovní zkušenosti i za konzultace při tvorbě diplomové práce.

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostaně a použil jsem pouze podklady uvedené v přiloženém seznamu.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon).

V Praze dne 13. 1. 2007 .....

Abstrakt

Diplomová práce se zabývá měřením svítidel a letištních návěstidel. Popisuje požadavky kladené na letištní návěstidla, způsob měření jejich vyzařovacích charakteristik a metodiku ověřování toho, zda vyhovují daným normám. V rámci práce byl navržen a implementován software s funkcemi potřebnými pro měření zdrojů světla a zpracování výsledných dat, část práce se věnuje implementaci tohoto software. Jedna kapitola je věnována implementaci firmware pro průmyslový mikropočítač, který slouží jako rozhraní pro komunikaci zúčastněných zařízení. Důležitou částí práce je popis metody pro obecný přepočet souřadnic mezi polohou otočného stolu goniofotometru s nedosažitelným středem polohování a pozorovacími úhly světelného zdroje.

Abstract

The Master's thesis addresess the field of measuring luminaires and aerodrome lights. It describes the requirements put on an aerodrome light, the method of measuring its luminous characteristics, and a way of ensuring that it complies with the specifications given. There was a software developed as a part of the thesis, which allows for measuring the luminaires and evaluating the resulting data. One chapter of the thesis describes the design and the implementation of that software. There is a chapter dedicated to the implementation of a firmware of a microcomputer which serves as a communication interface between participating devices. An important part of the thesis describes a general method for converting between the goniophotometer rotating table orientation and the observational angles of the luminaire.

Obsah

1 Úvod **1**

2 Popis měření **2**
2.1 Úvod . . . 2
2.2 ICAO a dokument Annex 14 . . . 2
2.3 Požadavky na letištní návěstidla . . . 2
2.3.1 Souřadná soustava letištních návěstidel. . . . 3
2.3.2 Rozdělení letištních návěstidel . . . 3
2.3.3 Všesměrová návěstidla . . . 3
2.3.4 Směrová návěstidla, izokandelové diagramy . . . 4
2.3.5 Návěstidla přibližovacího světelného systému . . . 4
2.3.6 Ostatní návěstidla . . . 5

3 Popis zkušebny a goniofotometru **7**
3.1 Použité fotometrické veličiny . . . 7
3.2 Goniofotometr . . . 7
3.2.1 Goniofotometr s pevným zdrojem světla a pohyblivým fotočlánkem . . . 7
3.2.2 Goniofotometr s pohyblivým zdrojem světla a pevným fotočlánkem . . . 8
3.2.3 Goniofotometr s rotujícími zrcadly . . . 8
3.3 Současné vybavení zkušebny . . . 8
3.4 Modernizace zkušebny . . . 9

4 Převod mezi polohou stolu a pozorovacími úhly **13**
4.1 Úvod . . . 13
4.2 Typografické konvence . . . 13
4.3 Definice pojmů . . . 14
4.4 Transformace při pohybu stolu . . . 18
4.5 Převod souřadnic stolu na pozorovací úhel . . . 19
4.5.1 Převod . . . 19
4.6 Převod pozorovacího úhlu na souřadnice stolu . . . 20
4.6.1 Zadání . . . 20
4.6.2 Implementace . . . 22
4.6.3 Testování . . . 22

5 Firmware jednotky CKDM **24**
5.1 Úvod . . . 24
5.2 Komunikace s PC . . . 25
5.2.1 Úvod . . . 25
5.2.2 Datový rámec . . . 25
5.2.3 Zotavení z chyb při přenosu . . . 26

5.2.4 Příkazy . . . 27
5.3 Komunikace po sběrnici IEEE 488.1 . . . 28
5.4 Ovládání luxmetru . . . 30
5.5 Ovládání otočného stolu . . . 30
5.6 Ovládání ramene . . . 31
5.7 Uživatelské rozhraní . . . 32
5.8 Podpůrné moduly . . . 33
5.8.1 ad.c, ad.h . . . 33
5.8.2 atol.c, atol.h . . . 33
5.8.3 block.c, block.h . . . 33
5.8.4 buzzer.c, buzzer.h . . . 33
5.8.5 cmos.c, cmos.h . . . 34
5.8.6 globals.h . . . 34
5.8.7 intvec.c . . . 34
5.8.8 main.c . . . 34
5.8.9 misc.c, misc.h . . . 34
5.8.10 mutex.h . . . 34
5.8.11 num2str.c, num2str.h . . . 34
5.8.12 p_num_x_x.c, p_num_x_x.h . . . 34
5.8.13 panel.c, panel.h . . . 35
5.8.14 parse_float.c, parse_float.h . . . 35
5.8.15 rct.c, rtc.h . . . 35
5.8.16 sleep.c, sleep.h . . . 35
5.8.17 timer.c, timer.h . . . 35
5.8.18 uart.c, uart.h . . . 35

6 Software **36**
6.1 Úvod . . . 36
6.2 Požadavky na aplikaci . . . 36
6.2.1 Definice typových zkoušek . . . 36
6.2.2 Odměr dat . . . 36
6.2.3 Vyhodnocování typových zkoušek . . . 36
6.2.4 Individuální vývoj . . . 37
6.2.5 Kalibrace stolu, ramene a luxmetru . . . 37
6.2.6 Ukládání dat . . . 37
6.2.7 Export a tisk protokolů . . . 37
6.2.8 Hlášení chyb . . . 37
6.2.9 Nastavení programu . . . 38
6.3 Volba programovacího jazyka . . . 38
6.3.1 Porovnání některých jazyků . . . 38
6.4 Použité externí knihovny . . . 39

6.5 Architektura aplikace . . . 40
6.6 Implementace vzoru Model View Controller . . . 41
6.7 Hlavní obrazovka . . . 45
6.8 Definice typových zkoušek . . . 45
6.8.1 Izokandelové typové zkoušky . . . 45
6.8.2 Polární typové zkoušky . . . 45
6.8.3 Tabulkové typové zkoušky. . . . 46
6.8.4 Implementace definic typových zkoušek . . . 46
6.9 Měření . . . 46
6.10 Vyhodnocování typových zkoušek . . . 47
6.11 Individuální vývoj . . . 48
6.11.1 Vývoj izokandelových návěstidel . . . 48
6.11.2 Vývoj všesměrových návěstidel a svítidel . . . 48
6.11.3 Vývoj tabulkových měření. . . . 48
6.12 Ukládání naměřených dat . . . 48
6.13 Tisk a export protokolů . . . 49
6.14 Komunikace s přístroji . . . 49
6.15 Převod mezi polohou stolu a pozorovacím úhlem . . . 49
6.16 Práce s grafy . . . 49
6.17 Ošetření chyb . . . 50
6.17.1 Chyby způsobené uživatelem . . . 50
6.17.2 Chyby programu . . . 51

7 Závěr **52**

8 Seznam literatury **53**

Rejstřík **55**

A Anglicko-český slovníček některých pojmů **56**

B Obsah přiloženého CD **57**

C Obrazové přílohy **58**

Seznam obrázků

1 Azimut, elevace . . . 3
2 Definice vyzařovací charakteristiky koncového dráhového návěstidla pomocí izokandelového diagramu . . . 4
3 Pohled pilota na přistávací dráhu vybavenou systémem indikace sestupové přibližovací roviny PAPI . . . 5
4 Schematický pohled na zkušebnu zeshora, před modernizací . . . 9
5 Polohovací stůl goniofotometru . . . 10
6 Stůl s přípravkem . . . 11
7 Schematický pohled na zkušebnu zeshora, po modernizaci . . . 12
8 Pohyb stolu a jeho umístění do absolutní souřadné soustavy . . . 15
9 Definice vektorů $\vec{O}$, $\vec{P}$, $\vec{A}$ . . . 16
10 Zavedená souřadná soustava . . . 17
11 Jednotka CKDM . . . 24
12 Struktura datového rámce pro komunikaci s PC . . . 25
13 Úvodní okno aplikace Goniofotometr . . . 36
14 Rozdělení aplikace Goniofotometr na jednotlivé moduly . . . 41
15 Odměřování kalibrační matice . . . 58
16 Hlášení chyby programu . . . 59
17 Vyhodnocení typové zkoušky, graf bez luminogramu . . . 59
18 Vyhodnocení typové zkoušky, graf s luminogramem . . . 60
19 Rozšíření měření při vývoji izokandelových návěstidel . . . 60
20 Výběr tématu luminogramu . . . 60
21 Příprava na individuální vývoj . . . 61
22 Stav připojených přístrojů . . . 61

1 Úvod

Společnost **ELTODO Power s.r.o.** poskytuje služby v oblasti výroby a dodávek v oboru elektro se zaměřením na veřejné osvětlení, silnoproudou a slaboproudou techniku, řídící a dopravní systémy a letištní světelná zařízení. Její divize **Elektrosignál** pak vyrábí a vyvíjí kompletní sortiment světelného zabezpečovacího zařízení pro letiště a heliporty mnoha států světa. Výrobky splňují podmínky I., II. a III. kategorie **ICAO** mezinárodního předpisu o civilním letectví. Pro prověřování a podporu vývoje svítidel a návěstidel má Elektrosignál vlastní zkušebnu. Ta obsahuje především goniofotometr – přístroj na měření svítivosti světelného zdroje. Cílem diplomové práce je modernizace a rozšíření možností právě této zkušebny.

Diplomová práce částečně navazuje na již realizovanou dizertační práci [Veselý97], v mnoha směrech ovšem rozšiřuje původní vybavení pracoviště. Nejdůležitější inovace nyní ve zkratce popíšu.

- Do zkušebny bude nainstalováno otočné rameno pro měření zdrojů světla, které se musí měřit ve vodorovné poloze, protože jejich vyzařovací charakteristiky jsou závislé na jejich orientaci. Pro otočné rameno byla přidána nová jednotka, CKDM, která slouží jako pult pro ruční ovládání ramene a zároveň jako rozhraní pro komunikaci mezi PC a všemi periferiemi.

- Luxmetr a otočný stůl komunikují po sběrnici GPIB. V původní verzi byla přímo v PC umístěna karta s GPIB řadičem. V nové verzi tato karta není potřeba, GPIB protokol byl implementován přímo v jednotce CKDM na GPIO výstupech procesoru. PC nyní komunikuje se stolem a s luxmetrem prostřednictvím jednotky CKDM přes rozhraní RS232.

- Software byl kompletně přepsán. Původní verze byla napsaná v programovacím prostředí Famulus, běžela v DOSU, grafický výstup zajišťovaly BGI ovladače firmy Borland. Nová verze běží pod OS Windows 98 a vyšší, je psána v jazyce Python. Uživatelské prostředí odpovídá dnešním zvyklostem, grafy a protokoly jsou exportovány do formátu PDF, všechna data se ukládají ve formátu XML. Jsou použity nativní Windows widgety a vše se dá ovládat jak klávesnicí, tak myší. Při vývoji nových návěstidel se dají potřebné izokandely, popisky a další prvky přesouvat pomocí drag and drop. Byly přidány izokandelové diagramy, které pomocí barvy indikují svítivost v tom kterém bodě naměřené mřížky.

- Přepočet souřadnic ze souřadné soustavy zdroje světla do souřadné soustavy stolu je analyticky neřešitelný problém. Původní DOSová verze spočítala souřadnice naklopení a natočení stolu nahrubo a poté naměřené výsledky interpolovala do požadovaných bodů. Nová verze implementuje nový způsob přepočítávání, který dříve nebyl možný pro nízký výpočetní výkon počítače PC AT 286. Souřadnice jsou numerickou metodou spočítány přesně a měří se přímo v požadovaných bodech.

2 Popis měření

2.1 Úvod

Na každém letišti je umístěno mnoho návěstidel, která slouží k různým účelům. Sestupová návěstidla informují pilota o tom, zda je ve správné výšce, mnoho druhů směrových i všesměrových návěstidel vyznačuje obrysy či osy přistávacích a pojezdových drah, další světla označují překážky, jiné zase informují o tom, zda je letová dráha volná nebo obsazená. Samozřejmě není myslitelné, aby každé letiště mělo vlastní systém osvětlení. Ten je standardizován v rámci dokumentů mezinárodní organizace **ICAO**.

2.2 ICAO a dokument Annex 14

O standardizaci leteckého provozu usiluje mezinárodní organizace **ICAO**[1]. ICAO byla založena v roce 1947 a je specializovaným oddělením OSN[2]. Mezi její hlavní cíle patří zabezpečování a výměna informací o leteckém provozu, ochrana životního prostředí, zabezpečování efektivnosti leteckého provozu a prosazování jednotných mezinárodních leteckých zákonů.

Předpisy pro osvětlení letišť a heliportů jsou popsány v dokumentu [**Annex**]. Ten předepisuje rozměry a vlastnosti letištních ploch a vlastnosti technického vybavení letišť včetně umístění a charakteristik návěstidel. Svazek I se věnuje letištím, svazek II heliportům. Poslední, čtvrté vydání svazku I bylo přijato radou ICAO 28. února 2004. Poslední, druhé vydání svazku II bylo přijato radou ICAO 14. března 1995.

2.3 Požadavky na letištní návěstidla

ICAO definuje jaké mechanické, elektroinstalační, poziční a fotometrické požadavky musí návěstidla splňovat. Požadavky jsou shrnuty v [Annex] v kapitole 5.3 (Lights). Přesné vyžadované charakteristiky jednotlivých návěstidel jsou potom popsány v [Annex] v příloze B (Aeronautical Ground Light Characteristics). Fotometrické požadavky se dají rozdělit na následující:

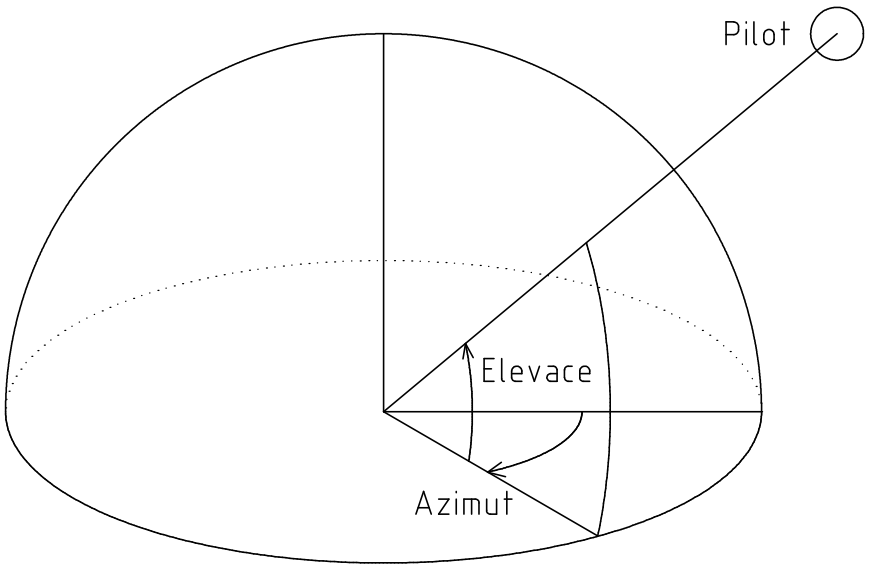
1. Správná barva (chromatické požadavky)
2. Minimální svítivost v definovaných obastech a bodech
3. Symetrie svítivosti v bílé a červené části sestupových návěstidel PAPI[3]
4. Rovnoměrnost svítivosti v hlavním svazku některých směrových návěstidel

Goniofotometr neumožňuje měřit barvu, chromatické požadavky jsou ovšem nejsnáze splnitelné. S pomocí goniofotometru se tedy vyhodnocují body 2 až 4.

[1] International Civil Aviation Organization
[2] Organizace spojených národů
[3] viz kapitola 2.3.5

2.3.1 Souřadná soustava letištních návěstidel.

Pro popis směru od návěstidla/k návěstidlu se používá dvojice souřadnic **azimut** a **elevace**. Azimut určuje pozorovací úhel ve vodorovné rovině. Orientace je taková, že při pohybu okolo návěstidla po směru hodinových ručiček azimut roste. Elevace určuje výšku nad návěstidlem. Orientace elevace je taková, že čím výše je pilot nad návěstidlem, tím větší je elevace.



Obrázek 1: Azimut, elevace

V izokandelových diagramech (viz dále) je použit stejný souřadný systém, souřadnice v diagramech tedy udávají pozorovací úhel návěstidla.

2.3.2 Rozdělení letištních návěstidel

Návěstidla se dají podle požadavků na svítivost v zásadě rozdělit na čtyři druhy druhy: všesměrová, směrová, sestupová a ostatní. Dále popíšu požadavky kladené na každý typ návěstidel. Uvedu pouze obecné požadavky na návěstidla, přesné definice lze dohledat v [Annex], kapitola 5. Izokandelové diagramy pro směrová a sestupová návěstidla jsou v [Annex], příloha 2.

2.3.3 Všesměrová návěstidla

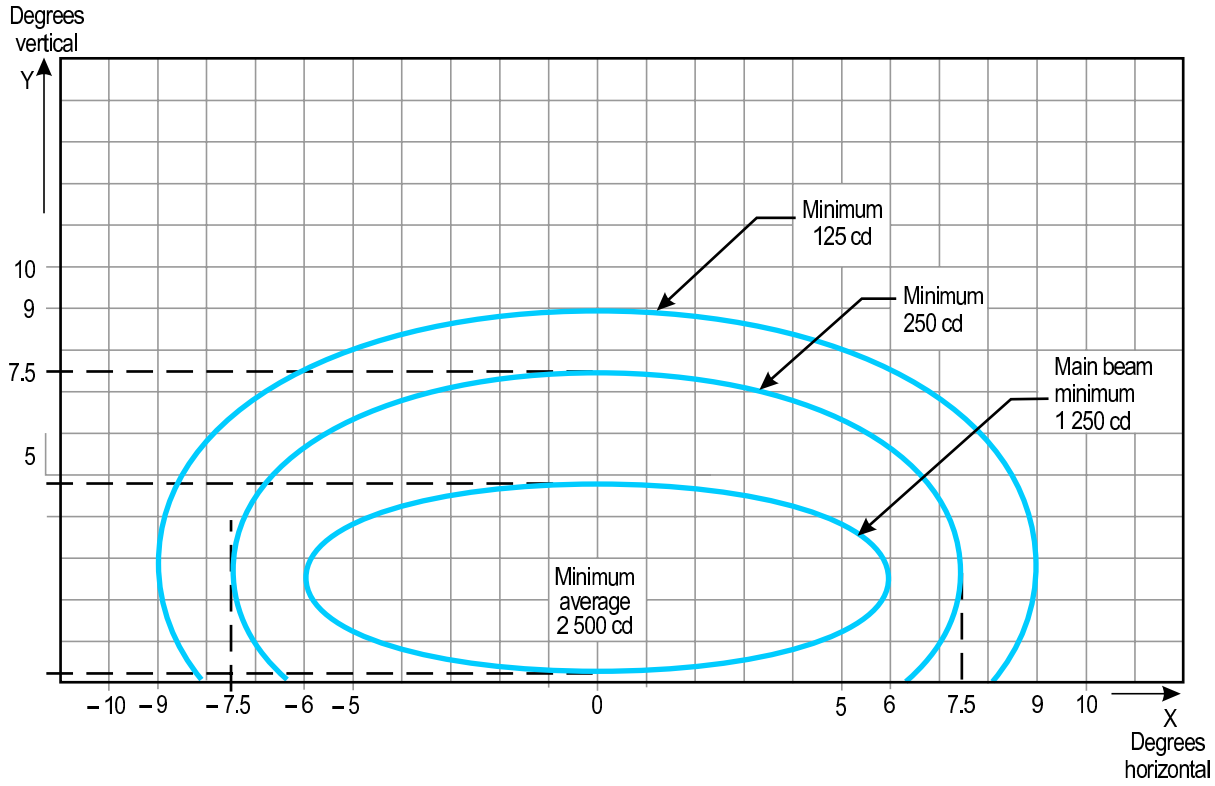
Všesměrová návěstidla musí vyzařovat ve všech hodnotách azimutu, většinou pro nějaké dané rozmezí elevace. Příkladem budiž návěstidla postranní dráhové řady, která jsou rozmístěna na okraji přistávací a vzletové dráhy. Pokud mají pomáhat navádět letadlo při kroužení, musí podle ICAO vyzařovat pro všechny úhly azimutu a pro nula až patnáct stupnů elevace. Pilot kroužící okolo letiště tak vidí dráhová postranní návěstidla ze všech stran. Mezi všesměrová návěstidla patří i překážková návěstidla, která upozorňují na vysoké budovy, věže, vedení apod. ve vzdušném prostoru. Ta samozřejmě také musí být viditelná ze všech stran.

Schvalování probíhá nejčastěji tak, že se pro několik daných úhlů elevace naměří svítivost pro všech 360° azimutu a pro azimut s nejnižší svítivostí se naměří svítivost pro elevace od 0° do 90°. Všechny naměřené body pak musí splňovat minimální předepsanou svítivost. Měření

všech 360.90 = 32400 bodů mřížky by při cca pěti sekundách na odměr bodu nebylo z časových důvodů možné, měří se tedy pouze rozsah elevace pro kritický azimut.

2.3.4 Směrová návěstidla, izokandelové diagramy

Směrová návěstidla vyzařují pouze v poměrně úzkém prostorovém svazku, vrcholový úhel vyzařovaného svazku je maximálně několik desítek stupňů. Příkladem směrového návěstidla jsou koncová dráhová návěstidla. Ta se umísťují na konec dráhy a červeným světlem pilotovi ukazují, kde dráha končí. Nemá ovšem smysl, aby svítila i v jiných směrech než směrem k pilotovi na dráze.



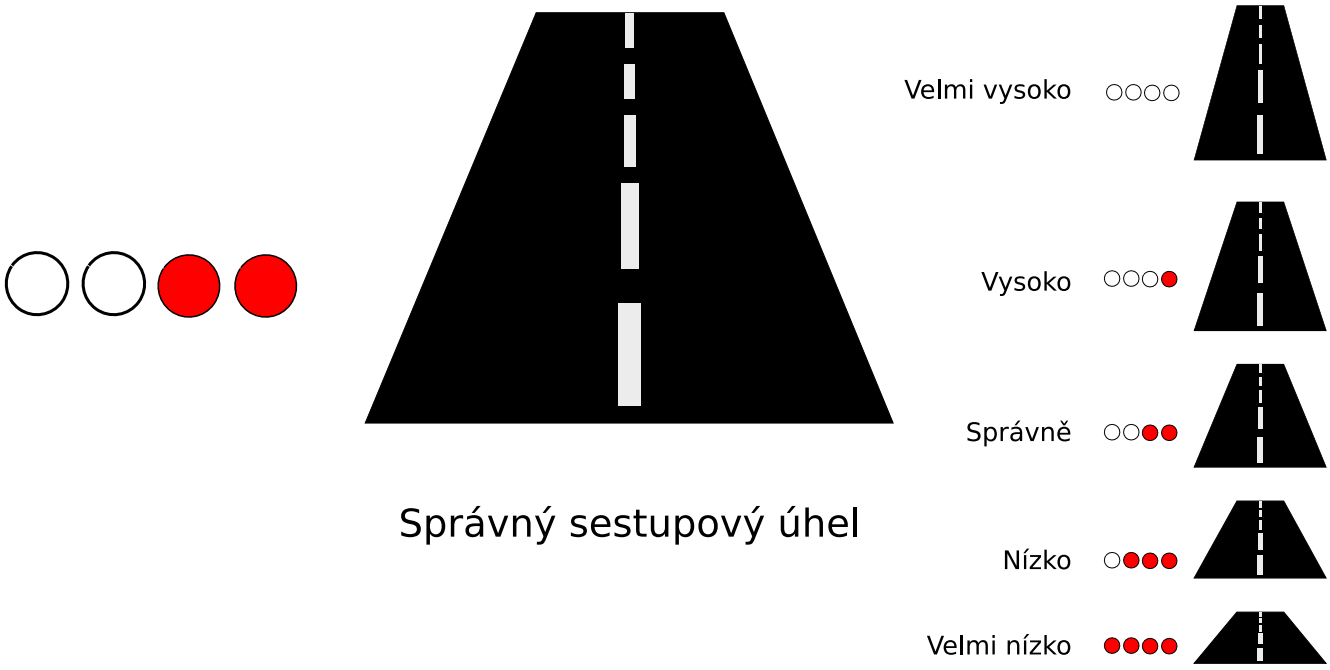
Obrázek 2: Definice vyzařovací charakteristiky koncového dráhového návěstidla pomocí izokandelového diagramu

Požadavky na směrová návěstidla jsou definovány pomocí tzv. **izokandelových diagramů.** Izokandelové diagramy popisují minimální požadovanou svítivost pro celé oblasti vyzařovaného světla. Oblasti jsou ohraničeny matematickými křivkami – elipsami, ovály, obdélníky, obdélníky se zkosenými rohy. Protože není technicky možné provést spojité měření přes všechny body dané oblasti, měří se pouze v bodech mřížky, nejčastěji s rozestupem 1° azimutu i elevace. Na vnitřní oblast, **hlavní svazek**, je kladen další požadavek – aritmetický průměr svítivosti ve všech bodech hlavního svazku nesmí být menší než definovaná hodnota.

2.3.5 Návěstidla přibližovacího světelného systému

Sestupová návěstidla slouží k vizuální navigaci pilota při přistávání letadla a pomáhají mu najít správný sestupový úhel. V praxi se používají sestupová návěstidla dvojího druhu: VASIS

(Visual Approach Slope Indicators) a PAPI (Precision Approach Path Indicator). Princip navádění je však u obou typů podobný. Návěstidla mají vertikálně ostře rozpůlenou vyzařovací charakteristiku, v jedné polovině svítí bíle a v druhé červeně. Pilot pak vidí různá návěstidla svítit červeně nebo bíle podle toho, zda je ve správné sestupové zóně, příliš vysoko nebo příliš nízko.



Obrázek 3: Pohled pilota na přistávací dráhu vybavenou systémem indikace sestupové přibližovací roviny PAPI

Na obrázku 3 je zobrazeno, jak vidí pilot přistávací dráhu a navigační systém PAPI (umístěný vedle dráhy) při různých sestupových úhlech. Pokud je příliš vysoko, vidí pouze bílou část všech návěstidel. Pokud je ve správné výšce, vidí bílou část dvou levých návěstidel a červenou dvou pravých návěstidel. Pokud je letadlo příliš nízko, vidí pilot všechna návěstidla přibližovacího systému svítit červeně.

Definice je podobná definici směrových návěstidel, charakteristika se také udává pomocí izokandelových diagramů a oblastí s definovanými minimálními svítivostmi. Minimální svítivost se ovšem udává zvlášť pro bílou část a pro červenou část. Průměrná svítivost hlavního svazku se neměří, ve svazku přesahujícím do bílé i červené oblasti by neměla dobrý fyzikální význam. Sestupová návěstidla PAPI ovšem musí splňovat další podmínku, svítivost libovolného bodu v bílé oblasti musí být přinejmenším dvounásobkem a maximálně 6,5 násobkem svítivosti odpovídajícího bodu v červené oblasti. Odpovídajícím bodem se zde rozumí bod se stejným azimutem, ale se zápornou elevací (symetrický podle přímky elevace, ve které se bílá charakteristika mění na červenou).

2.3.6 Ostatní návěstidla

ICAO popisuje i charakteristiky dalších návěstidel, ale z různých důvodů nemá význam dělat přesná měření goniofotometrem. Jedním takovým typem jsou zábleskové návěstidla, která

nesvítí stále, ale s frekvencí v řádu jednotek hertzů blikají. Luxmetr není schopen efektivní hodnotu svítivosti měřit. Další světla nebo návěstidla nejsou již přesně popsána, např. u semaforů je zadána pouze barva a směr světelného svazku bez přesnější specifikace. Zde může být užitečné naměřit charakteristiku návěstidla v některých vybraných bodech a tak se ujistit, že návěstidlo bude dobře plnit svoji funkci.

3 Popis zkušebny a goniofotometru

3.1 Použité fotometrické veličiny

Požadavky na vyzařovací charakteristiky zdrojů světla se udávají v jednotkách svítivosti, **kandelách** (zkratka cd). **Svítivost** je hustota světelného toku bodového zdroje v daném směru.

Luxmetr, snímač použitý v goniofotometru, měří **osvětlenost**, což je veličina vyjadřující světelný tok dopadající na určitou plochu. Jednotkou osvětlenosti je jeden **lux** (zkratka lx), osvětlení způsobené světelným tokem o svítivosti jeden lumen dopadajícím na plochu jeden metr čtvereční.

Osvětlenost lze snadno převést na svítivost (intenzitu světla) podle vzorce $I = E.r^2$, kde E[lm] je osvětlenost v lumenech, I[cd] je svítivost v kandelách a r[m] je vzdálenost snímače osvětlenosti od zdroje světla.

3.2 Goniofotometr

Hlavním přístrojem pro měření návěstidel je ve zkušebně Elektrosignál přístroj zvaný **goniofotometr**. Goniofotometr je přístroj, který dokáže měřit svítivost zdroje světla z různých pozorovacích úhlů. Základním prvkem goniofotometru je vždy snímač luxmetru – fotočlánek, který měří osvětlenost. Aby bylo možné automatizovaně měřit svítivost z různých pozorovacích úhlů, je vždy součástí goniofotometru nějaký pohyblivý prvek. Podle konstrukčního řešení pohybu se dají nejpoužívanější goniofotometry rozdělit do tří skupin.

3.2.1 Goniofotometr s pevným zdrojem světla a pohyblivým fotočlánkem

Zdroj světla se umístí na pevnou podložku a dále se s ním nepohybuje. Okolo zdroje světla se pak pohybuje rameno s fotočlánkem.

Výhodou tohoto typu goniofotometru je, že světelný zdroj lze umístit přímo do středu polohování ramene a nastavený úhel ramene pak přímo odpovídá pozorovacímu úhlu. Tento typ je také výhodný pro měření světelných zdrojů, jejichž vyzařovací charakteristiky jsou závislé na orientaci. Například výbojková světla jsou podélně symetrická a otočení takového světla okolo podélné osy příliš nemění vyzařovací charakteristiku, ale klopení takového světla již může výsledky významně ovlivnit.

Nevýhodou je především malá vzdálenost zdroje od fotočlánku. Pro měření svítivosti je nutné, aby byl zdroj světla bodový. Žádný reálný zdroj světla samozřejmě bodový není, ale čím větší je vzdálenost snímače od zdroje, tím více se zdroj z pohledu snímače blíží ideálnímu. Sestupová letištní návěstidla je nutné měřit ze vzdálenosti minimálně 12 metrů, konstrukčně je ovšem téměř nemožné vyrobit goniofotometr s požadovanou přesností (řádově úhlové minuty) a s pohyblivým ramenem o poloměru 12 metrů.

3.2.2 Goniofotometr s pohyblivým zdrojem světla a pevným fotočlánkem

Zdroj světla je umístěn na pohyblivém stole se dvěma stupni volnosti. Fotočlánek se nepohybuje, pozorovací úhel závisí pouze na orientaci stolu se zdrojem světla.

V tomto případě u většiny konstrukčních řešení nelze umístit zdroj světla přímo do os polohování stolu, úhly nastavené na měřicím stole tedy neodpovídají přesně pozorovacím úhlům. Zdroj světla se totiž při pohybu stolu nejen natáčí a naklápí, ale i pohybuje. Další nevýhodou je, že na tomto typu goniofotometru není možné korektně měřit zdroje světla, jejichž světelný tok se mění v závislosti na orientaci, například výbojky.

Hlavní výhodou, zvláště pro měření letištních návěstidel, je možnost umístit snímač luxmetru do značné vzdálenosti od zdroje světla. Pokud prostor zkušebny dovolí, lze umístit fotočlánek prakticky neomezeně daleko. Pro potlačení vlivu okolního a odraženého světla je nutné zakrýt okolí optické trasy černou látkou s pokud možno co nejnižší světelnou odrazivostí, to se ovšem týká všech typů goniofotometrů. Další optické clony na cestě světelného svazku odrážejí světlo směrem pryč od fotočlánku.

Tento typ goniofotometru je použit ve zkušebně Elektrosignál.

3.2.3 Goniofotometr s rotujícími zrcadly

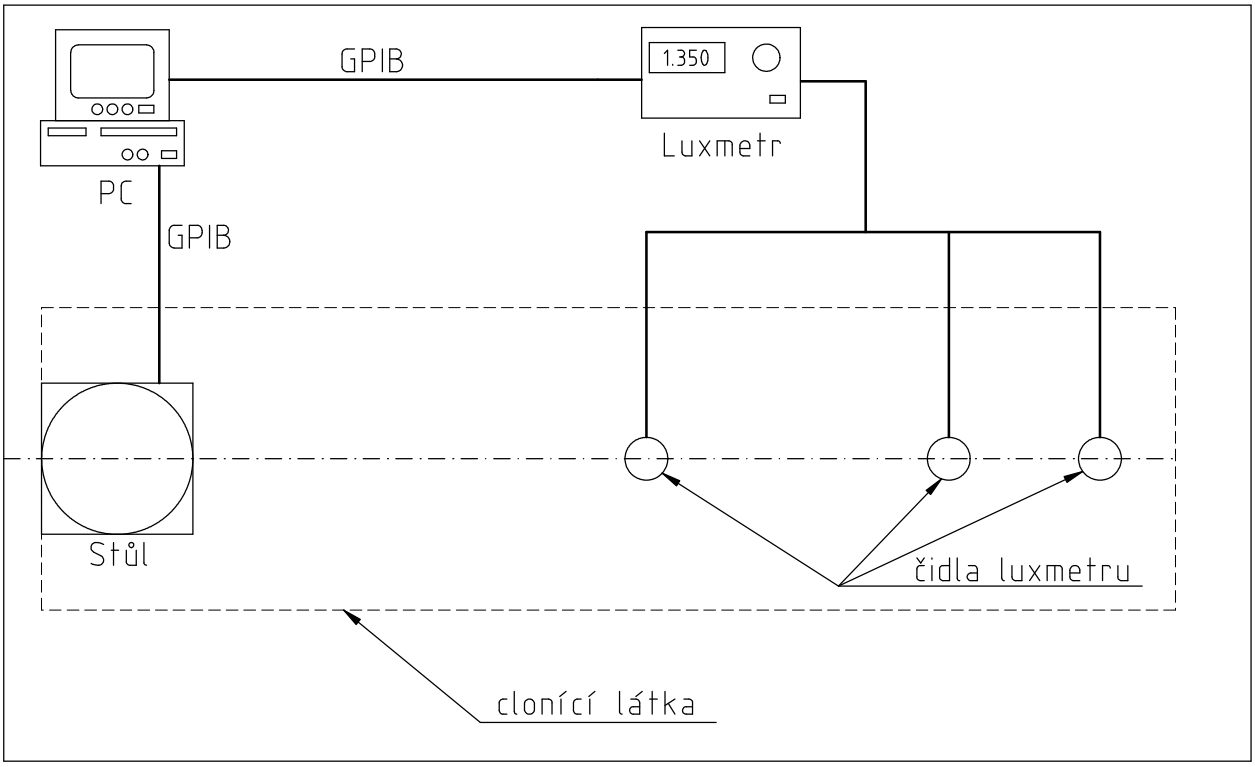
Tento typ goniofotometru má nepohyblivý zdroj světla i fotočlánek. Okolo zdroje světla se pohybuje zrcadlo se dvěmi stupni volnosti a odráží světlo zdroje do snímače, který může být i velmi vzdálený. Tento typ goniofotometru kombinuje výhody předchozích dvou typů, je ovšem velmi obtížné vyrobit jej s dostatečnou přesností. Nejkvalitnější vyráběné goniofotometry dosahují přesnosti nastavení pozorovacích úhlů asi 0.1°. Také clony v optické trase musí mít větší průměr, protože zdánlivý zdroj světla se pohybuje po větším prostoru spolu se zrcadlem, výrazněji se tedy uplatní vliv odraženého záření.

3.3 Současné vybavení zkušebny

Ve zkušebně je umístěn goniofotometr se dvěma stupni volnosti, s pohyblivým zdrojem světla a pevným fotočlánkem (viz 3.2.2), počítač PC AT 286 a jehličková tiskárna. Počítač komunikuje s otočným stolem a luxmetrem po sběrnici GPIB, pro tento účel je ve sběrnici ISA počítače zasunutá speciální karta s GPIB řadičem. Schematický pohled propojení přístrojů při pohledu na zkušebnu shora je ukázán na obrázku 4.

Goniofotometr ve zkušebně sestává z otočného stolu se dvěma stupni volnosti a z luxmetru.

Stůl je vyrobený na zakázku firmou Mikronex s.r.o. Ve skříni je oboustranně upevněna naklápěcí část stolu s rozsahem ±45°. Horní část naklápěné části je otočná, rozsah otáčení je ±180°. Přesnost polohování v natáčení i v naklápění je $0,01°$při maximální nosnosti až 40 kg. Tato přesnost je pro měření letištních návěstidel nutná, například ostrost přechodu mezi bílým a červeným pásmem sestupových soustav PAPI nesmí být vyšší než tři úhlové minuty. Stůl lze ovládat buď lokálně z panelu, nebo vzdáleně po sběrnici GPIB. Pro měření všesměrových návěstidel lze navíc připevnit na stůl tzv. **přípravek**.



Obrázek 4: Schematický pohled na zkušebnu zeshora, před modernizací

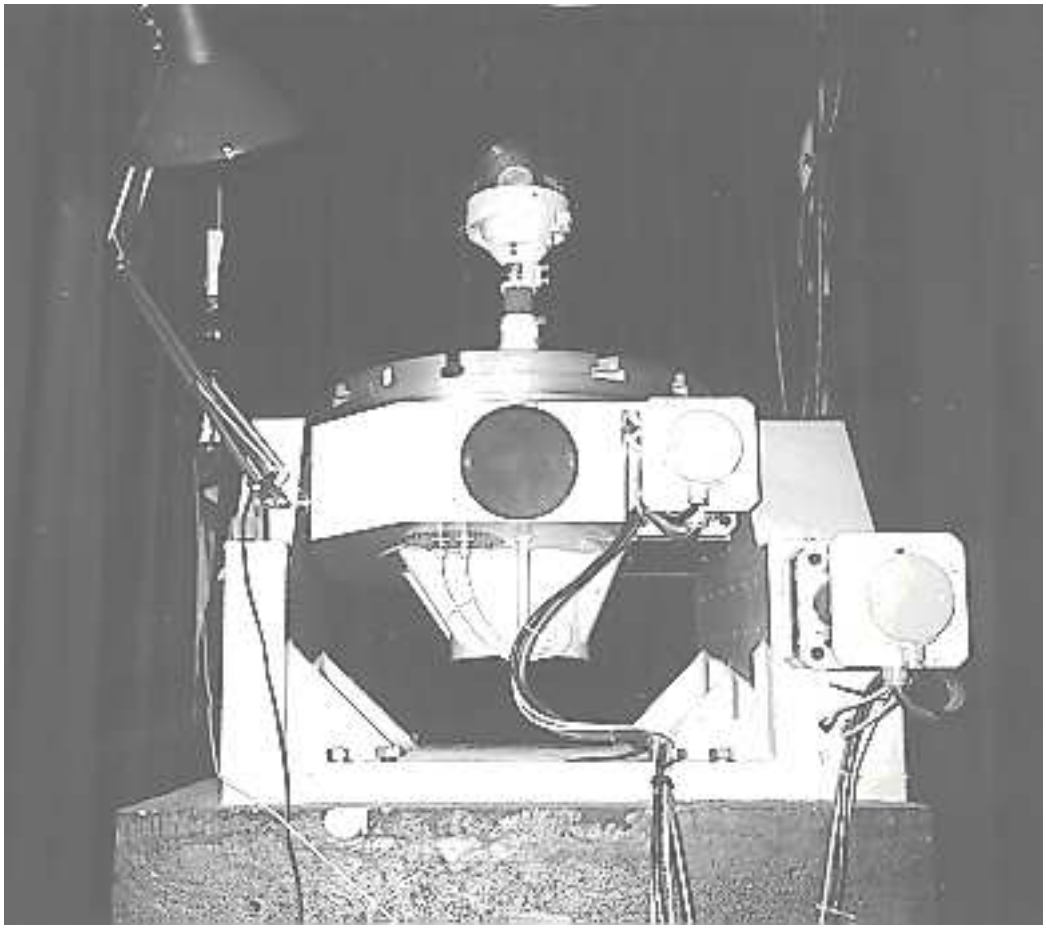
Přípravek vypadá jako klín se sklonem 45°, který se připevní na upínací plochu stolu. Převod v přípravku převádí rotační pohyb upínací plochy stolu na rotační pohyb upínací plochy přípravku, celý měřený zdroj světla se tak vlastně naklopí o dalších 45°. Jako vedlejší efekt ovšem přípravek obrací smysl otáčení upínací plochy. Schematicky je stůl s přípravkem zobrazen na obrázku 6

Jako luxmetr je použit přístroj LTM550B firmy LMT LICHTMESSTECHNIK GMBH. Původně byl určen pouze pro ruční ovládání, ale byl upraven tak aby dokázal komunikovat i s dalšími zařízeními. Nyní komunikuje po sběrnici GPIB, při zapnutém lokálním ovládání je možné ručně přepínat rozsahy. Právě měřenou hodnotu průběžně zobrazuje na displeji. Snímač luxmetru se umísťuje do předem připravených držáku v definovaných vzdálenostech[1].

3.4 Modernizace zkušebny

Jedním z cílů modernizace je umožnit měření svítidel v pracovní poloze, tedy s vodorovně orientovanou podélnou osou. Za tímto účelem se bude do zkušebny instalovat otočné rameno, které se bude nad stolem pohybovat spolu s upevněným fotočlánkem. Jako řadič krokového motoru byla vybrána programovatelná jednotka CD30x firmy MICROCON s.r.o., komunikující po rozhraní RS232. Staré PC AT 286 bude nahrazeno moderním stolním počítačem. Komunikaci mezi stolem, luxmetrem, ramenem a PC bude zajišťovat jednotka CKDM-11/12 (dále jen CKDM) firmy ELSACO Kolín, která bude zároveň sloužit i jako panel pro lokální ovládání ramene. Schéma propojení jednotlivých komponent je vidět na obrázku 7. V následujících odstavcích podrobněji popíši nové komponenty.

[1] 2,980 m, 5,807 m, 15,810 m



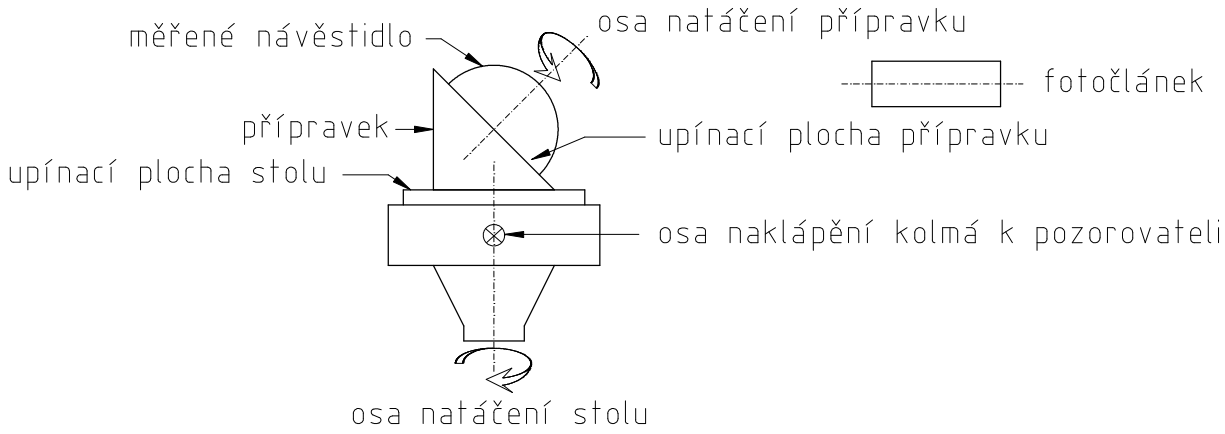
Obrázek 5: Polohovací stůl goniofotometru

Stolní počítač. Záměrem je provozovat software na běžném kancelářském počítači. Požadavky pro rozumný běh software jsou: taktovací frekvence CPU > 1GHz, velikost RAM 256MB a více, 200MB volného místa na disku, volný sériový port. Software běží pod OS Windows 98 a vyšší, pro prohlížení a tisk PDF protokolů je nutné mít nainstalovaný Adobe Reader. Naprostá většina současných kancelářských PC tyto požadavky bez problémů splňuje. Důležité je, že již není třeba speciální karta s GPIB řadičem, veškerá komunikace s periferiemi probíhá z pohledu PC přes rozhraní RS232. Některé současné počítače již nemají integrovaný sériový port, není ovšem problém koupit např. převodník USB⇒RS232.

Rameno. Rameno je poháněno krokovým motorem SX34-3080 firmy MICROCON s.r.o. přes převodovku s poměrem 1:100. Krokový motor ovládá programovatelná jednotka CD30x téže firmy, konkrétně řadič M1486. Jednotka CD30x komunikuje s nadřízenou jednotkou (v tomto případě CKDM) přes rozhraní RS232.

Jednotka CKDM. Jednotka CKDM firmy ELSACO Kolín je průmyslový ovládací terminál s membránovou klávesnicí, znakovým displejem s rozlišením 20×4 znaků, dvěma řadiči rozhraní RS232, mnoha GPIO[2] vývody, osmi A/D převodníky a mnoha dalšími periferiemi. Srdcem jednotky je mikroprocesor MB90598G ze série F^{2}MC-16 firmy FUJITSU s taktem až 16MHz, 128KB paměti ROM/flash (data, kód) a 4KB paměti RAM. Jednotku CKDM jsme vybrali z

[2]General Purpose Input/Output

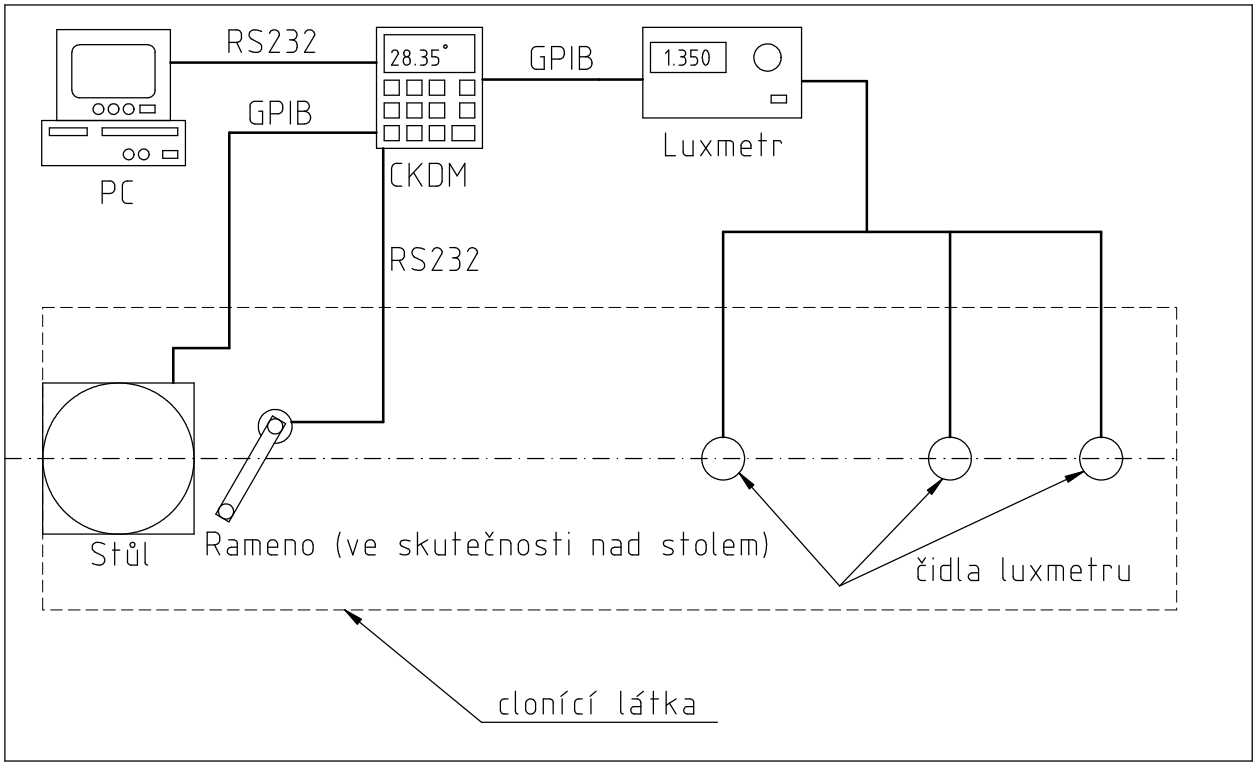


Obrázek 6: Stůl s přípravkem

několika důvodů:

- S touto jednotkou mám několikaleté zkušenosti, věděl jsem co od ní očekávat a nemusel jsem se příliš bát nenadálých problémů.
- Na A/D převodník lze připojit potenciometr testující, zda se rameno skutečně pohybuje. Nějaká překážka mohla zastavit pohyb ramene, proto je na hřídel připojen i potenciometr, který se otáčí spolu s hřídelí. Podle polohy potenciometru lze zhruba určit, zda je rameno v poloze odpovídající nastaveným souřadnicím.
- CKDM má vyvedených mnoho GPIO vstupů/výstupů pro volné použití. Z velké části jsou využity na připojení signálů sběrnice GPIB. Komunikaci po této sběrnici jsem musel implementovat ručně, potřeba je minimálně 19 vývodů (8×data, 8×kontrolní signály, 3×čtení signálů DAV, NRFD, NDAC).
- Rozlišení displeje je dostatečné pro zobrazování potřebných informací – úhlu ramene, chybových hlášek apod. Prvních 8 znaků je programovatelných, čehož jsem využil pro znaky jako je ° (stupeň) a hodiny (čekání).
- Membránová klávesnice je odolná, tichá a klávesy se dobře tisknou.
- Malé rozměry umožňují bez problému umístit CKDM do panelu na zdi.
- Dva kanály RS232 umožňují komunikovat zároveň s PC i s řadičem krokového motoru MH1386.
- Při rozhodování svoji roli hrála i cenová dostupnost jednotky CKDM.

Ostatní komponenty – stůl a luxmetr – zůstávají a jsou používány i po modernizaci.



Obrázek 7: Schematický pohled na zkušebnu zeshora, po modernizaci

4 Převod mezi polohou stolu a pozorovacími úhly

4.1 Úvod

Goniofotometr použitý ve zkušebně nemá dosažitelný střed polohování, to znamená že nelze umístit světelný zdroj přímo do průsečíku os natáčení a naklápění. Nepříjemným důsledkem je, že si přesně neodpovídají pozorovací úhly – azimut a elevace (viz 2.3.1) a úhly naklopení a natočení stolu, protože se světelný zdroj pohybuje zároveň s pohybem stolu. Analyticky lze přesně spočítat azimut a elevaci při známém natočení a naklopení, ale opačný výpočet je nemožný. Stará verze software počítala souřadnice natočení a naklopení přibližně a naměřená data následně interpolovala do požadovaných bodů. Hlavní nevýhodou tohoto postupu byla samozřejmě nově zavedená chyba měření, neměřilo se ve skutečně požadovaných bodech, ale v jejich blízkém okolí. Chyba určení souřadnic stolu byla tím větší, čím blíže byl snímač a čím dále bylo návěstidlo umístěno od středu polohování.

V nové verzi goniofotometru jsem pro hledání přesných souřadnic natočení a naklopení použil numerickou metodu. Nová metoda je zcela obecná, umožňuje vzájemné převody souřadných soustav stolu a světla při libovolném umístění zdroje světla, fotočlánku i počáteční poloze stolu. Tento postup nebylo možné ve staré verzi použít pro příliš nízký výpočetní výkon počítače PC AT 286, hledání souřadnic by výrazně zpomalilo celé měření. Na současných počítačích ovšem počítání souřadnic trvá kratší dobu než samotné měření, takže lze počítat souřadnice v jednom vlákně vícevláknového programu a samotné měření provádět v jiných vláknech. Měření není zpomaleno počítáním souřadnic a zároveň je možné zjistit přesné hodnoty natočení a naklopení stolu pro odměřování dalších bodů.

V této kapitole popíši problematiku převodu mezi souřadnou soustavou světla a souřadnou soustavou stolu a konkrétní postup, který jsem použil. Převod souřadných soustav jsem řešil zvlášť pro tři různé případy:

1. obyčejné měření bez přípravku
2. měření s přípravkem
3. měření s ramenem

Důvodem pro toto dělení je, že při měřeních typu 2. a 3. je matematika jednodušší, zdroj světla se totiž vždy umísťuje do osy natáčení a při natáčení stolu se tedy nemění jeho poloha. Vzájemný obousměrný převod mezi souřadnou soustavou světla a stolu je v těchto případech vyjádřitelný analyticky. V této kapitole se budu věnovat pouze převodu typu 1.

4.2 Typografické konvence

Body jsou značeny velkým písmenem, např. bod B. Vektory jsou značeny velkým písmenem se šipkou, např. vektor $\overrightarrow{V}$. Přímky dané dvěma body jsou značeny dvojicí těchto bodů s pruhem, např. $\overline{AB}$. Vektory definované dvěma body, počátečním bodem X a koncovým bodem Y jsou

značeny $\overrightarrow{XY}$. Roviny jsou definovány třemi body, které v rovině leží, takto definované roviny jsou značeny XYZ. Rovina také může být definována dvěma body a vektorem, který je s rovinou rovnoběžný, taková rovina je značena $XY\vec{V}$. Osy souřadné soustavy jsou značeny x, y, z.

4.3 Definice pojmů

Pro definici pohybu stolu a převodu souřadných soustav budu pracovat v euklidovském prostoru $\mathbb{E}^3$. V tomto prostoru je umístěn stůl a jeho osy, měřený zdroj světla i snímač luxmetru – fotočlánek. Pro popis polohy a orientace stolu, zdroje světla a fotočlánku použiji následující pojmy:

1. Absolutní souřadná soustava: Euklidovský prostor $\mathbb{E}^3$. Střed polohování stolu je v bodu $[0, 0, 0]$ prostoru $\mathbb{E}^3$, osa naklápění stolu splývá s osou x, při nulovém naklopení stolu osa otáčení upínací desky stolu splývá s osou z.

2. Úhly σ, ω: Úhly naklopení, resp. natočení stolu. Úhel σ – naklopení stolu – se pohybuje v rozmezí $\pm 45°$. Úhel ω – otočení stolu – se pohybuje v rozmezí $\pm 180°$. Na obrázku 8. a) až 8. c) je znázorněn pohyb stolu a jeho umístění do absolutní souřadné soustavy. Osa naklápění splývá s osou x, kladný smysl naklápění stolu je levotočivý vůči ose x. Osa otáčení splývá s osou z, kladný smysl otáčení je pravotočivý vůči ose z.

3. Nulová poloha stolu: Poloha, ve které se nachází stůl když $\sigma = 0$, $\omega = 0$. Upínací deska stolu je v nulové poloze orientována vodorovně. Stůl v nulové poloze je zobrazen na obrázku 8. c).

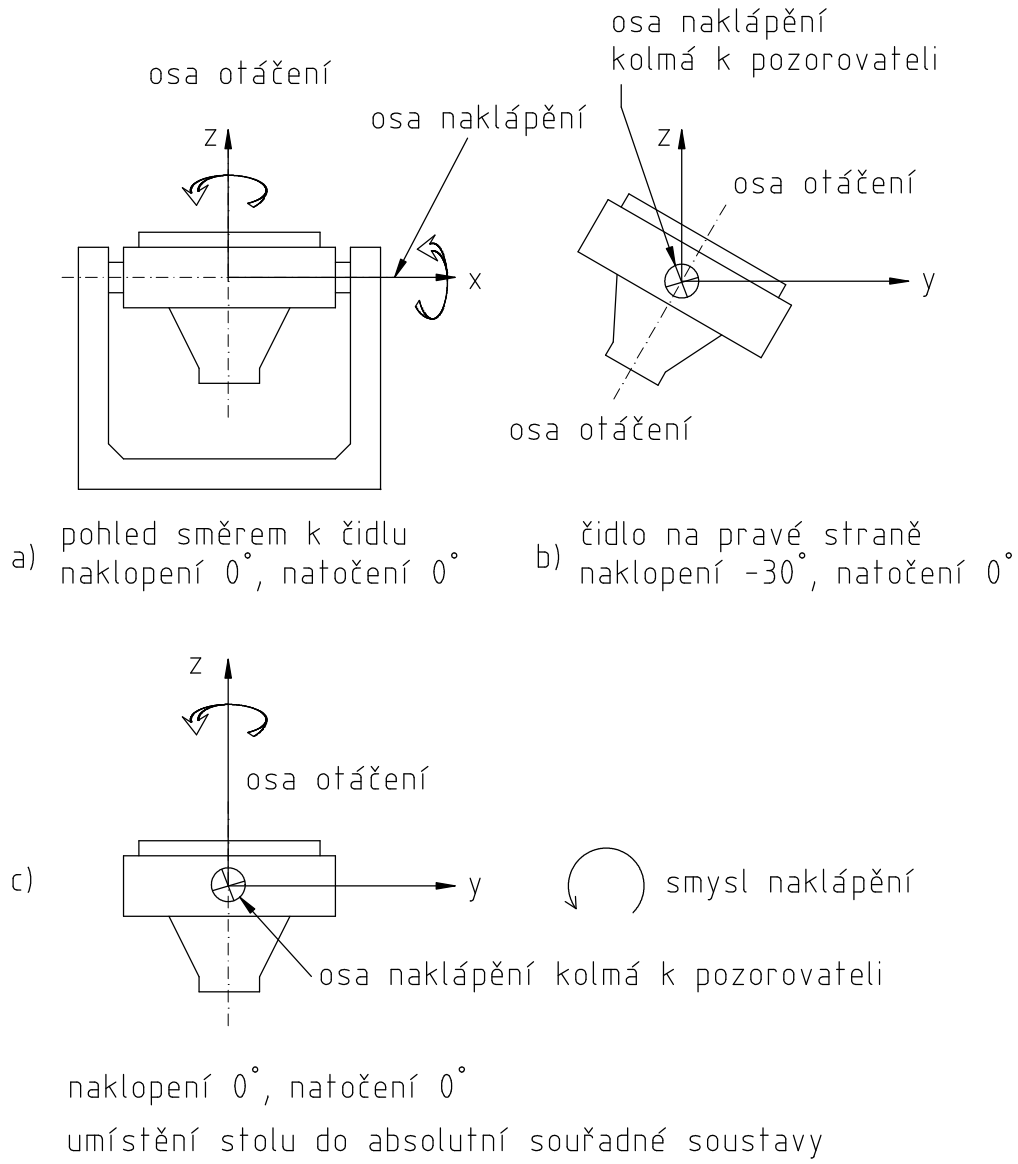
4. Střed polohování stolu: Průsečík os naklápění a natáčení stolu. V prostoru $\mathbb{E}^3$má souřadnice $[0, 0, 0]$.

5. Bod S: Tento bod určuje polohu snímače, jeho poloha se v průběhu výpočtu nemění.

6. Bod F: Souřadnice ohniska zdroje světla v nulové poloze stolu. Skutečné svítidlo má samozřejmě nenulovou velikost, ale zde si je nahradím ideálním bodovým zdrojem světla, umístěným v F. Při pohybu stolu se vždy transformuje poloha bodu F, zdroj světla nelze umístit do středu polohování stolu. Obsluha goniofotometru zadává polohu ohniska vzhledem k upínací ploše stolu, ale přepočet na F je velmi snadný, stačí k z-ové souřadnici polohy ohniska přičíst vzdálenost upínací plochy od středu polohování stolu.

7. Vektor $\overrightarrow{FS}$: Určuje pozici snímače vzhledem k ohnisku světelného zdroje.

8. Úhly θ, ϕ: Tyto úhly reprezentují pozorovací úhel zdroje světla. Při měření letištních návěstidel odpovídá azimut úhlu θ a elevace úhlu ϕ. Jde vlastně o variantu úhlů sférické souřadné soustavy.



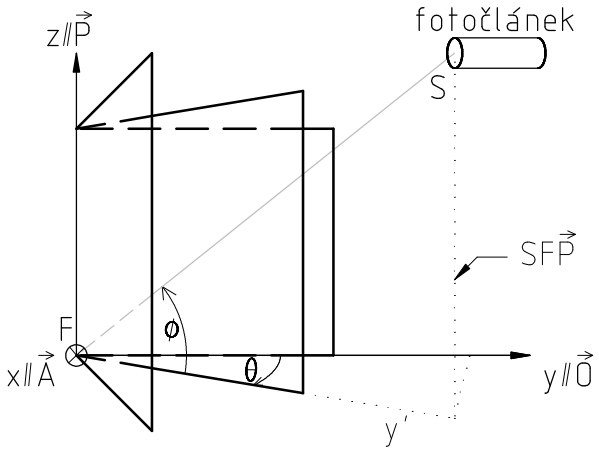
Obrázek 8: Pohyb stolu a jeho umístění do absolutní souřadné soustavy

9. Vektory $\vec{O}$, $\vec{P}$, $\vec{A}$: Tyto tři jednotkové navzájem kolmé vektory definují orientaci návěstidla. Vektor $\vec{O}$ určuje směr vyzařování světelného toku. Vektor $\vec{P}$ je rovnoběžný s průsečíkem svazku rovin procházejících F, které definují úhel θ. Vektor $\vec{A}$ se použije při určování úhlu θ. Na obrázku 9 je zobrazen příklad jednoduché situace, kde ohnisko F je umístěno do středu prostoru $\mathbb{E}^3$, vektor $\vec{O}$ splývá s osou y, vektor $\vec{P}$ splývá s osou z a vektor $\vec{A}$ splývá s osou x. Pokud si představíme, že rovina xy reprezentuje zemi a směr osy z je směrem stoupání letadla, vidíme klasickou situaci, kde snímač luxmetru S – pilot – pozoruje ze vzduchu letištní návěstidlo F. Úhel mezi rovinou $SF\vec{P}$ a rovinou yz definuje azimut – úhel θ. Promítneme-li osu y v pravoúhlém promítání do roviny $SF\vec{P}$, získáme její obraz y'. Úhel mezi přímkou y' a přímkou FS je elevace – úhel ϕ.

Příklad obecné situace umístění zdroje světla a snímače je na obrázku 10.

Definice: Funkce $arctan2(x, y) \to \mathbb{R}, x \in \mathbb{R}, y \in \mathbb{R}, (x, y) \in \mathbb{R}^2 - \{0, 0\}$ je definována takto:

- $arctan2(x, y) = \pi/2$ pro $x = 0, y > 0$



Obrázek 9: Definice vektorů $\vec{O}$, $\vec{P}$, $\vec{A}$

- $arctan2(x, y) = -\pi/2$ pro $x = 0, y < 0$
- $arctan2(x, y) = \arctan(y/x)$ pro $x > 0$
- $arctan2(x, y) = \arctan(y/x) - \pi$ pro $x < 0$ •

Funkce arctan2 je velmi podobná funkci arctan, ale je korektně definovaná i pro vektory s nulovou nebo zápornou x-ovou souřadnicí.

Definice: Úhel od vektoru k vektoru okolo vektoru. Jsou dány tři nenulové vektory $\vec{A}$, $\vec{B}$, $\vec{X}$, přičemž ani vektor $\vec{A}$, ani vektor $\vec{B}$ nejsou násobkem vektoru $\vec{X}$. Existuje taková transformace T zachovávající úhly a délky[1], že $T(\vec{X}) = [0, 0, q]$, $q > 0$. Aplikací této transformace na vektory $\vec{A}$, $\vec{B}$ získáme jejich obrazy $\vec{A}'$, $\vec{B}'$. Nyní spočtěme souřadnici ϕ pro body $A' = \left[\overrightarrow{A'_x}, \overrightarrow{A'_y}, \overrightarrow{A'_z}\right]$ a $B' = \left[\overrightarrow{B'_x}, \overrightarrow{B'_y}, \overrightarrow{B'_z}\right]$ vyjádřené v cylindrických souřadnicích:

$$A_\phi = arctan2\left(A'_x, A'_y\right)$$

$$B_\phi = arctan2\left(B'_x, B'_y\right)$$

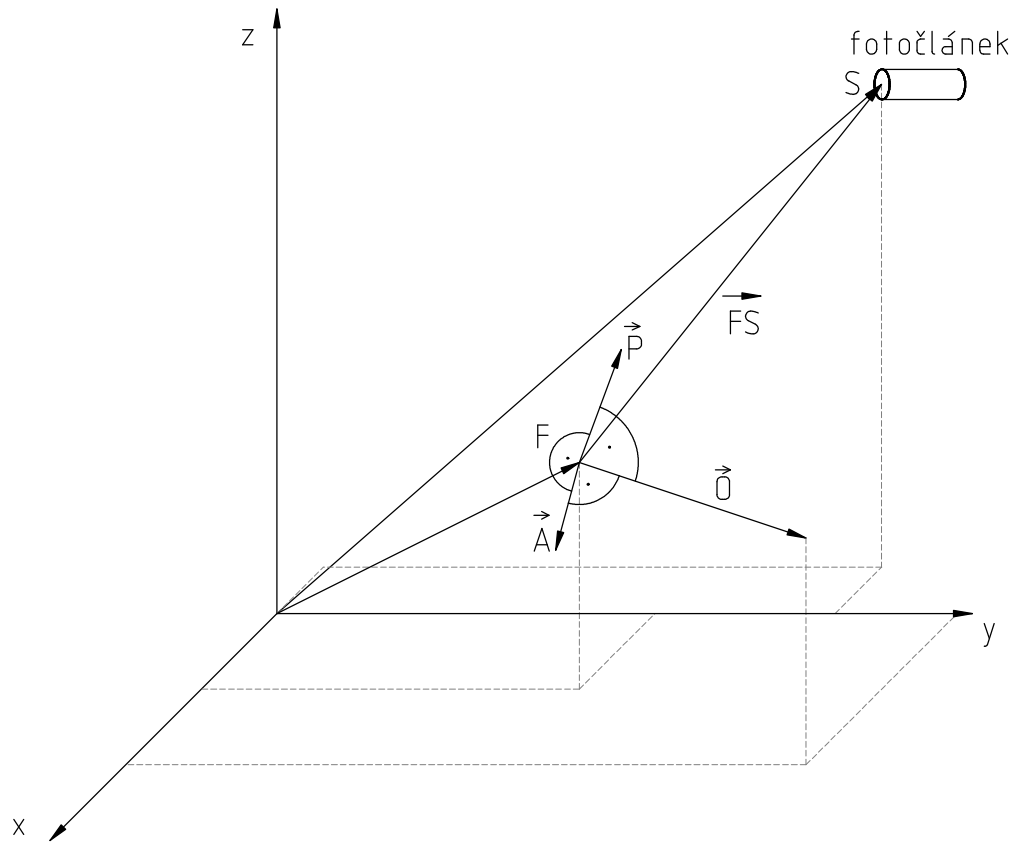
Úhel od vektoru $\vec{A}$ k vektoru $\vec{B}$ okolo vektoru $\vec{X}$ je úhel $A_\phi - B_\phi$ •

Úhel od vektoru $\vec{A}$ k vektoru $\vec{B}$ okolo vektoru $\vec{X}$ lze zjistit např. tímto způsobem:

$$\phi' = \arccos \frac{(\vec{A} \times \vec{X}).(\vec{B} \times \vec{X})}{|\vec{A} \times \vec{X}||\vec{B} \times \vec{X}|}$$

$$\phi = \begin{array}{l} \phi' \text{ pro } ((\vec{A} \times \vec{X}) \times (\vec{B} \times \vec{X})).\vec{X} < 0 \\ -\phi' \text{ pro } ((\vec{A} \times \vec{X}) \times (\vec{B} \times \vec{X})).\vec{X} \geq 0 \end{array}$$

[1] Takové vlastnosti má transformace otočení.



Obrázek 10: Zavedená souřadná soustava

Úhel ϕ je pak úhel od vektoru $\vec{A}$ k vektoru $\vec{B}$ okolo vektoru $\vec{X}$.

Definice: Vektor vzniklý rotací vektoru okolo vektoru o úhel. Jsou dány dva nenulové vektory $\vec{A}$, $\vec{X}$ a úhel α. Existuje taková transformace T zachovávající úhly a délky, že $T(\vec{X}) = [0, 0, q]$, $q > 0$. Aplikací této transformace na vektor $\vec{A}$ získáme jeho obraz $\vec{A'}$. Spočtěme nyní cylindrické souřadnice bodu $A' = \left[\overrightarrow{A'}_x, \overrightarrow{A'}_y, \overrightarrow{A'}_z\right]$

$$\begin{aligned} A'_\phi &= arctan2\left(A'_x, A'_y\right) \\ A'_r &= \sqrt{\left(A'_x\right)^2 + \left(A'_y\right)^2} \end{aligned}$$

Souřadnice A'_z se nemění. Zaveďme nyní nový bod B' v cylindrických souřadnicích takto:

$$\begin{aligned} B'_\phi &= A'_\phi - \alpha \\ B'_r &= A'_r \\ B'_z &= A'_z \end{aligned}$$

Tento bod má v kartézské soustavě následující souřadnice:

$$\begin{aligned} B'_x &= B'_r \cos\left(B'_\phi\right) \\ B'_y &= B'_r \sin\left(B'_\phi\right) \end{aligned}$$

souřadnice B'_z zůstává.

Nyní aplikujme na vektor $\left[B'_x, B'_y, B'_z\right]$ transformaci T^{-1} inverzní k transformaci T, získáme jeho obraz $\vec{B}$. Vektor $\vec{B}$ je **vektor vzniklý rotací vektoru $\vec{A}$ okolo vektoru $\vec{X}$ o úhel α.** •

Postup výpočtu souřadnic vektoru vzniklého rotací vektoru okolo vektoru o úhel zde nebudu uvádět. Při implementaci v software jsem použil hotové vzorce z [rotace].

4.4 Transformace při pohybu stolu

Představme si, že máme stůl v nulové poloze a k němu pevně připevněné návěstidlo F. Zajímá nás poloha návěstidla po otočení stolu o ω stupňů a naklopení o σ stupnů. Pohyb bodu F lze rozdělit na dvě nezávislé fáze. Nejdříve provedeme levotočivou rotaci okolo osy z (osa otáčení stolu v nulové poloze) o úhel ω a získáme bod F'. Poté provedeme pravotočivou rotaci okolo osy x (osa naklápění stolu) o úhel σ a získáme požadovaný transformovaný bod F''. Pohyb bodu lze popsat pomocí transformačních matic.

Matice natočení $R(\omega)$ pro daný úhel natočení ω:

$$R(\omega) = \begin{bmatrix} \cos\omega & \sin\omega & 0 \\ -\sin\omega & \cos\omega & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Pro bod F' platí:

$$F' = R(\omega).F$$

Matice naklopení $T(\sigma)$ pro daný úhel naklopení σ (pozor, narozdíl od matice R je pravotočivá):

$$T(\sigma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\sigma & -\sin\sigma \\ 0 & \sin\sigma & \cos\sigma \end{bmatrix}$$

Pro výsledný bod F'' platí:

$$F'' = T(\sigma).F'$$

Obě matice lze složit do výsledné transformační matice $M(\omega, \sigma)$:

$$M(\omega,\sigma) = T(\sigma).R(\omega) = \begin{bmatrix} \cos\omega & \sin\omega & 0 \\ \sin\omega\cos\sigma & \cos\omega\cos\sigma & -\sin\sigma \\ -\sin\omega\sin\sigma & \cos\omega\sin\sigma & \cos\sigma \end{bmatrix}$$

pak platí:

$$F'' = M(\omega,\sigma).F$$

Může se stát, že při upevňování návěstidla není stůl v nulové poloze, ale v nějaké obecné

poloze ω, σ. Obsluha se snaží návěstidlo upevnit tak, aby směr šíření paprsků (vekor $\vec{O}$) byl rovnoběžný s osou y. To může být snazší v případě, že je stůl pootočený nebo naklopený. V tomto případě nás bude zajímat přesun bodu při pohybu stolu do nulové polohy. Jde vlastně o inverzní pohyb vzhledem k výše popsanému pohybu. Nejprve musím provést naklopení o $-\sigma$ a poté otočení o $-\omega$.

$$T_z(\sigma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\sigma & \sin\sigma \\ 0 & -\sin\sigma & \cos\sigma \end{bmatrix}$$

$$F' = T_z(\sigma).F''$$

$$R_z(\omega) = \begin{bmatrix} \cos\omega & -\sin\omega & 0 \\ \sin\omega & \cos\omega & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$F = R_z(\omega).F'$$

$$M_z(\omega,\sigma) = T_z(\sigma).R_z(\omega) = \begin{bmatrix} \cos\omega & -\sin\omega\cos\sigma & -\sin\omega\sin\sigma \\ \sin\omega & \cos\omega\cos\sigma & \cos\omega\sin\sigma \\ 0 & -\sin\sigma & \cos\sigma \end{bmatrix}$$

$$F = M_z(\omega,\sigma).F''$$

4.5 Převod souřadnic stolu na pozorovací úhel

4.5.1 Převod

Pro stůl v nulové poloze je zadána poloha zdroje světla F_0, jeho orientace pomocí nenulových, navzájem kolmých vektorů $\vec{O}_0$, $\vec{P}_0$, $\vec{A}_0$, poloha snímače Sa úhly natočení a naklopení stolu ω, σ. Veškeré hodnoty mohou být voleny libovolně. Cílem je zjistit pozorovací úhly θ a ϕ.

Nejprve transformujme polohu návěstidla F_0 a vektory $\vec{O}_0, \vec{P}_0, \vec{A}_0$ s pomocí matice $M(\omega,\sigma)$ definované v části 4.4

$$\begin{aligned} F &= M(\omega,\sigma).F_0 \\ \vec{O} &= M(\omega,\sigma).\vec{O}_0 \\ \vec{P} &= M(\omega,\sigma).\vec{P}_0 \end{aligned}$$

$$\vec{A} \;=\; M(\omega,\sigma).\vec{A}_0$$

Nyní musí platit, že bod S neleží na přímce procházející bodem F a rovnoběžné s vektorem $\vec{P}$. Pokud bod S na této přímce leží, není úhel θ definovaný, stejně jako není definovaná např. zeměpisná délka na severním nebo jižním pólu. V tomto případě bude vždy buď $\phi = 90°$, nebo $\phi = -90°$.

S využitím definic z části 4.3 nyní lze vypočítat pozorovací úhly následujícím způsobem:

- **Úhel θ je úhel od vektoru $\vec{O}$ k vektoru $\overrightarrow{FS}$ okolo vektoru $\vec{P}$.**
- Nyní získejme vektor $\overrightarrow{FS'}$ rotací vektoru $\overrightarrow{FS}$ okolo vektoru $\vec{P}$ o úhel $-\theta$. **Úhel ϕ je úhel od vektoru $\vec{O}$ k vektoru $\overrightarrow{FS'}$ okolo vektoru $\vec{A}$.**

Pro letištní návěstidla lze ztotožnit úhel θ s azimutem a úhel ϕ s elevací např. v následující situaci:

- Směrové návěstidlo je umístěno v pracovní poloze, vyzařování hlavního svazku se děje ve směru osy y.
- $\vec{O} = [0, 1, 0]$
- $\vec{P} = [0, 0, 1]$
- $\vec{A} = [-1, 0, 0]$

4.6 Převod pozorovacího úhlu na souřadnice stolu

4.6.1 Zadání

Pro stůl v nulové poloze je zadána poloha zdroje světla F_0, jeho orientace pomocí nenulových, navzájem kolmých vektorů $\vec{O}_0$, $\vec{P}_0$, $\vec{A}_0$, poloha snímače S a požadované pozorovací úhly θ, ϕ. Veškeré hodnoty mohou být voleny libovolně. Cílem je zjistit takové úhly naklopení a natočení stolu ω, σ, pro které jsou pozorovací úhly shodné s požadovanými.

Tato úloha již není v obecném případě analyticky řešitelná, vede na soustavu rovnic s neznámými v argumentech goniometrických funkcí. Přesto je však nutné pro daný pozorovací úhel nějak „zjistit", do jaké polohy je třeba stůl uvést, aby skutečný pozorovací úhel odpovídal požadovanému. V zásadě jsou dvě možnosti, jak celou situaci řešit.

1. Řešit nějakou jednodušší úlohu, například umístit F do středu polohování. Pro F blízká středu polohování a snímač S relativně vzdálený od F a s nulovou hodnotou souřadnice S_x není chyba určení souřadnic příliš velká, hodnota svítivosti v požadovaném bodě pak může být interpolována z hodnot ve skutečně naměřených bodech.

2. Na řešení úlohy použít některou z numerických metod a iterativně nalézt úhly s dostatečně nízkou odchylkou, v případě goniofotometru s nižší odchylkou než je přesnost polohování stolu.

Zvolil jsem variantu č. 2, přičemž z numerických metod jsem vybral simplexovou metodu[2]. Ta umožňuje pomocí postupného přesouvání $n+1$ bodů maximalizovat nebo minimalizovat funkci n proměnných. V případě goniofotometru je třeba minimalizovat funkci určující odchylku požadovaných pozorovacích úhlů od úhlů spočtených v dané iteraci simplexového algoritmu.

V definici minimalizační funkce budeme potřebat následující definici:

Definice: **pozorovací vektor zdroje světla v základní poloze pro dané úhly.** Zadány jsou pozorovací úhly ω, σ. Proveďmě transformaci vektoru $\overrightarrow{V_0} = [1, 0, 0]$ maticí $M(\omega, \sigma)$:

$$\vec{V} = M(\omega, \sigma).\overrightarrow{V_0}$$

Vektor $\vec{V}$ je **pozorovací vektor zdroje světla v základní poloze pro úhly** ω, σ. •

Intuitivně lze chápat pozorovací vektor následovně: Zdroj světla umístěný ve středu polohování stolu a orientovaný tak, že vektor $\vec{O}$ ukazuje v kladném směru osy y, vektor $\vec{P}$ ukazuje v kladném směru osy z a vektor $\vec{A}$ ukazuje v kladném směru osy x pozorovatel v bodě $[\overrightarrow{V_x}, \overrightarrow{V_y}, \overrightarrow{V_z},]$ sleduje pod pozorovacími úhly ω, σ.

Nyní můžeme přikročit k samotné definici minimalizační funkce.

Definice: Jsou dány vektory určující orientaci návěstidla v nulové poloze stolu $\vec{O}_0$, $\vec{P}_0$, $\vec{A}_0$, poloha návěstidla v nulové poloze stolu F_0, poloha snímače S, požadované pozorovací úhly θ_p, ϕ_p a úhly natočení a naklopení stolu ω, σ.

Výsledek minimalizační funkce $f(\vec{O}_0, \vec{P}_0, \vec{A}_0, F_0, S, \theta_p, \phi_p, \omega, \sigma) \to \mathbb{R}$ pro zadané parametry získáme takto:

1. Pokud jsou souřadnice ω, σ mimo povolený rozsah polohování stolu, je

$$f(\vec{O}_0, \vec{P}_0, \vec{A}_0, F_0, S, \theta_p, \phi_p, \omega, \sigma) = \pi$$

2. Algoritmem popsaným v části 4.5.1 zjistíme pozorovací úhly θ, ϕ odpovídající nastavení stolu ω, σ, poloze zdroje světla F, orientaci zdroje světla dané hodnotami $\vec{O}_0, \vec{P}_0, \vec{A}_0$ a poloze snímače S.

3. Nechť $\overrightarrow{V_1}$ je pozorovací vektor zdroje světla v základní poloze pro pozorovací úhly θ, ϕ a $\overrightarrow{V_2}$ je pozorovací vektor zdroje světla v základní poloze pro pozorovací úhly θ_p, ϕ_p. Pro minimalizační funkci platí:

$$f(\vec{O}_0, \vec{P}_0, \vec{A}_0, F_0, S, \theta_p, \phi_p, \omega, \sigma) = \arccos\left(\overrightarrow{V_1}.\overrightarrow{V_2}\right)$$

•

Intuitivně minimalizační funkce vyjadřuje úhel mezi dvěma směry, kdy jeden směr představuje pilot pozorující návěstidlo z požadovaných pozorovacích úhlů, a druhý směr pilot pozorující návěstidlo ze stejných pozorovacích úhlů jako snímač luxmetru při daném postavení stolu.

[2]Též zvaná Downhill simplex method, Nelder-Mead method nebo Amoeba. Podrobnější popis lze nalézt např. v [recipes], kapitola 10.4

4.6.2 Implementace

Výsledkem běhu simplexové funkce jsou souřadnice stolu ω, σ a výsledek minimalizační funkce v nalezených souřadnicích f_{min}. Z hodnoty výsledku lze snadno zjistit, zda jde o požadovaný úhel, nebo pouze o lokální minimum minimalizační funkce. Minimalizační funkce je totiž zdola omezená, nejnižší možná hodnota je $f_{min} = 0$, tedy nulový úhel mezi nalezeným a požadovaným pozorovacím vektorem. Pokud je hodnota f_{min} vyšší než je povolená tolerance, simplexová metoda nalezla pouze lokální minimum, nebo úloha nemá řešení[3].

To, zda simplexová metoda nalezne globální nebo pouze lokální minimum, významně závisí na volbě výchozího bodu algoritmu. V software jsem implementoval následující algoritmus:

1. Nejprve se zjistí hodnoty minimalizační funkce pro $\omega = -180°, -150°, -120°, \ldots, 150°, 180°$ a pro $\sigma = -45°, -15°, 15°, 45°$. Tyto body se setřídí od nejnižšího k nejvyššímu podle hodnoty minimalizační funkce.

2. Postupně od bodu s nejnižší hodnotou minimalizační funkce se pro každý bod pouští běh simplexové metody. Pokud některý běh najde globální minimum, tedy příslušná $f_{min} < \epsilon$, kde ϵ je zvolená tolerance, je algoritmus úspěšně u konce. Pokud se vyčerpají všechny body, algoritmus skončil neúspěchem. Buď pro daný pozorovací úhel neexistuje platná poloha stolu, nebo algoritmus tuto polohu nenašel. Algoritmus neumožňuje rozhodnout, která varianta nastala.

4.6.3 Testování

Algoritmus uvedený v kapitole 4.6.2 jsem automatizovaně testoval pro několik tisíc různých zadání. Pro testování jsem použil následující algoritmus:

1. Náhodně vygeneruj jednotkové, navzájem kolmé vektory $\vec{O_0}, \vec{P_0}, \vec{A_0}$ tak, aby platilo: $\vec{O_{0y}} > 0$,

2. Náhodně vygeneruj pozici snímače S tak, aby $-1 < S_x < 1$, $2 < S_y < 20$, $-1 < S_z < 5$.

3. Vygeneruj pozici ohniska F_0 tak, aby $|F_0| < 1$.

4. Vygeneruj náhodně pozici stolu tak, aby $-180° < \omega < 180°$, $-45° < \sigma < 45°$

5. Pro dané uspořádání zjisti pozorovací úhly θ, ω

6. Pomocí simplexové metody pro uspořádání $F_0, S, \vec{O_0}, \vec{P_0}, \vec{A_0}$ a pozorovací úhly θ, ϕ se pokus najít příslušnou polohu stolu ω, σ.

Díky zadání je vždy zaručeno, že požadovanou polohu stolu lze nalézt. Volba rozsahů jednotlivých hodnot více než pokrývá potřeby zkušebny.

[3]Ne každé zadání úlohy má řešení. Stůl má omezený rozsah polohování a některé pozorovací úhly skutečně jsou nedosažitelné.

Tímto postupem jsem hledal polohu stolu pro několik tisíc zadání. Pro většinu bodů byla nalezena správná poloha stolu již v prvním běhu, v několika procentech případů až ve druhém nebo pozdějším běhu. Správnou polohu stolu se podařilo najít vždy.

5 Firmware jednotky CKDM

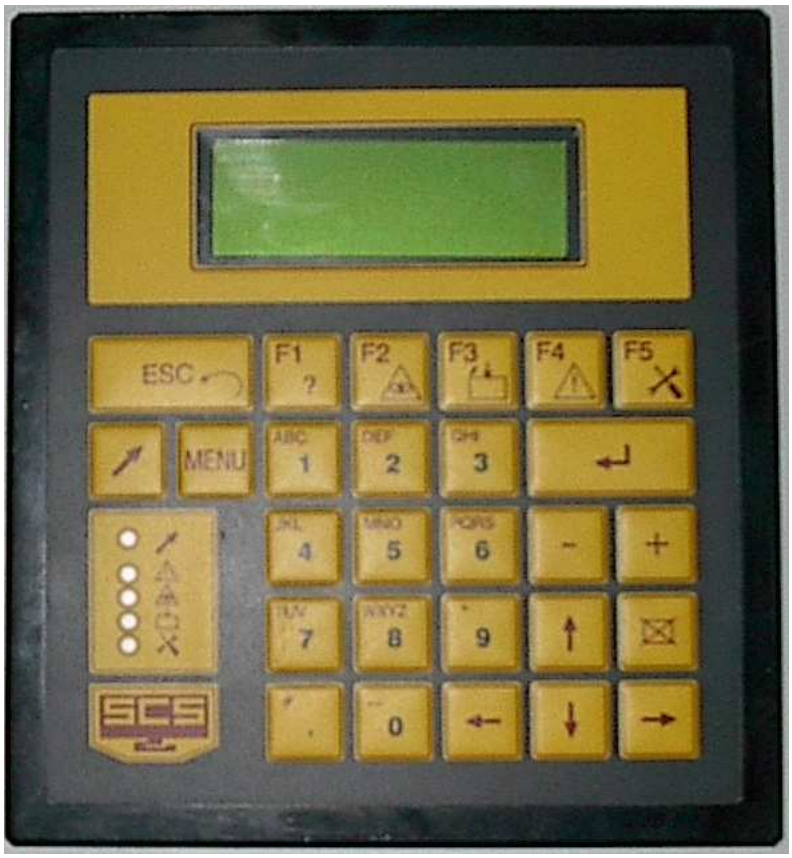
5.1 Úvod

Krátký popis jednotky CKDM a jejích možností je uveden v kapitole 3.4.

Firmware jednotky CKDM plní následující funkce:

- Komunikace s PC přes rozhraní RS232.
- Komunikace se stolem a s luxmetrem přes sběrnici GPIB.
- Komunikace s ramenem, resp. s řadičem krokového motoru CD30x přes rozhraní RS232.
- Panel pro ruční ovládání ramene.

Jednotka CKDM tedy slouží dvěma nezávislým účelům, jednak jako panel pro ruční ovládání ramene, druhak jako prostředník pro komunikaci PC a dalších periferií připojených přes sběrnici GPIB. Protože mnoho měření probíhá bez použití ramene, je na panelu jednotky CKDM umístěn vypínač napájení jednotky CD30x a výkonových stupňů motoru ramene. Pokud je vypnutý, slouží CKDM pouze jako interfejs pro komunikaci s luxmetrem a stolem, pokus o ovládání ramene končí chybou komunikace.



Obrázek 11: Jednotka CKDM

Firmware jednotky CKDM sestává ze zhruba 8 000 řádků v jazyce C, nemalou část jsem ovšem psal již dříve jen jako podpůrné moduly použité i v dalších projektech, které jsem s

CKDM realizoval. Kód je překládán pomocí překladače Softune C Compiler, poté je nahrán do flash paměti přes sériový port aplikací SKWizard.

Vše co má CKDM umět již bylo vyzkoušeno přímo ve zkušebně na potřebných přístrojích. Rameno zatím není fyzicky nainstalováno (zabetonováno do zdi), propojení **CKDM ⇒ CD30x ⇒ rameno** je ovšem funkční a odzkoušené. Chybí pouze potenciometr kontrolující polohu ramene [1].

V této kapitole postupně popíši nejdříve komunikaci jednotky CKDM s PC, poté komunikaci po sběrnici IEEE 488.1, ovládání stolu a luxmetru po této sběrnici, dále komunikaci s otočným ramenem a zvolené uživatelské rozhraní. Úplně nakonec uvedu krátký popis jednotlivých podpůrných modulů.

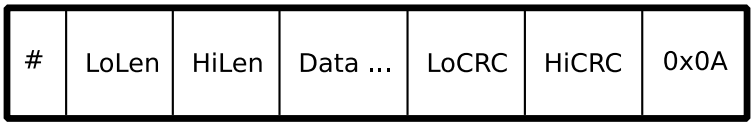
5.2 Komunikace s PC

5.2.1 Úvod

Bylo nutné vymyslet a implementovat protokol pro komunikaci s nadřízeným počítačem. Rozhodl jsem se pro potvrzovaný protokol zabezpečený kontrolním součtem, kdy jedině PC může poslat příkaz a jednotka CKDM na něj vždy odpoví právě jednou. Takový protokol je jednoduchý na implementaci a lze v něm snadno realizovat zotavení z chyb při přenosu. Implementace protokolu je v souborech `sercom.c, sercom.h`. Modul využívá pouze moduly `timer.c, timer.h, uart.c, uart.h a globals.h`, které jsou všechny psány tak aby šly přeložit pod OS Windows i pro CKDM. I samotný modul `sercom.c, sercom.h` lze tedy přeložit pro obě platformy, což je výhodné zejména při ladění programu.

5.2.2 Datový rámec

Data mezi dvěmi účastníky komunikace proudí jako ucelené datové rámce. Formát datového rámce je stejný pro příkaz i pro odpověd. Struktura jednoho rámce je nakreslena na obrázku 12. Popíši jednotlivá pole rámce:



Obrázek 12: Struktura datového rámce pro komunikaci s PC

- **#:** Znakem # (ASCII 35) každý rámec povinně začíná.
- **LoLen, HiLen:** Tyto dva bajty reprezentují nižší a vyšší bajt délky datového rámce. Do délky se započítávají pouze samotná data.
- **Data:** Len po sobě jdoucích bajtů, zde jsou uložena vlastní aplikačně závislá data.

[1] viz kapitola 5.8.1

- **LoCRC, HiCRC:** Nízký a vysoký bajt 16-bitového kontrolního součtu (CRC16). Kontrolní součet používá referenční implementaci publikovanou v [crc]
- **0x0A:** Znak nové řádky, musí být povinně na konci každého datového rámce a ukončuje jej.

V modulu `sercom` jsou funkce které usnadňují balení a rozbalování často používaných typů do/z datového rámce. Snažil jsem se udělat funkci pro posílání rámce co nejjednodušší na používání. Pro naprostou většinu situací stačí variadická funkce `sercom_outf()`, inspirovaná knihovní funkcí `printf()`. Prototyp funkce je:

```
int sercom_outf(const char *format, ...);
```

V proměnné `format` jsou postupně vypsány formáty jednotlivých variadických parametrů funkce. Výhodou je, že jednotlivá data se rovnou posílají na sériový port, není potřeba si na již tak velmi omezeném zásobníku připravovat místo pro celý buffer.

Lze balit typy

- uint32 (zkratka **i**), 32-bitový bezznaménkový integer
- int32 (zkratka **I**), 32-bitový integer se znaménkem
- uint32 (zkratka **c**), v programu se převádí z/na typ uint32, ale může nabývat pouze hodnot v rozmezí 0 .. 255, přenáší se jako jeden bajt
- char * (zkratka **s**), nulou ukončený řetězec

Pokud chci vyslat například řetězec "`Hello`", následovaný bezznaménkovým 32-bitovým číslem `42`, mohu použít následující volání:

```
sercom_outf("si", "Hello", 42);
```

Pro rozbalování slouží funkce `sercom_scanf`(), která pracuje na stejném principu.

5.2.3 Zotavení z chyb při přenosu

Komunikace vždy probíhá tak, že nadřízená jednotka (master) pošle příkaz, podřízená jednotka (slave) jej vykoná a pošle odpověď. Není možné, aby podřízená jednotka bez vyzvání sama začala posílat data. Komunikace se při dodržení určitých podmínek vždy zotaví po chybě při přenosu dat. Princip přenosu je tento:

1. Nadřízená jednotka pošle příkaz podřízené jednotce a čeká nejdéle čas $T_{master-timeout}$ na odpověď. Pokud podřízená jednotka do této doby neodpoví, může nadřízená jednotka zahlásit chybu nebo vysílání opakovat. Moje implementace opakuje vysílání třikrát a teprve poté hlásí chybu.
2. Podřízená jednotka má povinnost odpovědět na korektně přijatý příkaz nejdéle za čas $T_{slave-timeout}$ od příjmu posledního bajtu zprávy. Pro časy čekání na příkaz a odpověd platí : $T_{slave-timeout} = T_{master-timeout}/2$. Pokud přijdou chybná data, podřízená jednotka vůbec neodpovídá, čeká že nadřízená jednotka bude příkaz opakovat.

3. Příjem příkazu je shodný pro oba účastníky komunikace. Jakmile přijde první bajt, zkontroluje se zda jde o počáteční znak '#'. Pokud ne, příjem se přeruší a vyprázdní se vstupní buffer sériového portu. Poté se přijme délka, data, kontrolní součet a ukončovací znak. Pokud neodpovídá kontrolní součet nebo ukončovací znak, příjem se opět přeruší a vyprázdní se vstupní buffer. Mezi každými dvěma bajty přijímaných dat nesmí být prodleva delší než je $T_{timeout}$, jinak se opět přeruší příjem a vyprázdní vstupní buffer.

4. Vyprázdnění vstupního bufferu probíhá tak, že se zahazují všechny přijaté bajty, dokud přichází na port v prodlevách kratších než je $T_{timeout}$. Po uplynutí tohoto času bez příjmu bajtu je buffer považován za vyprázdněný.

Při dodržení tohoto protokolu se komunikace vždy zotaví při chybě přenosu. Při přenosu přikazu z nadřízené jednotky může dojít k následujícím situacím:

1. Zpráva je přijata korektně. Podřízená jednotka provede příkaz a pošle odpověď nejdéle za čas $T_{slave-timeout}$.

2. Zpráva vůbec nedorazí. Podřízená jednotka dále naslouchá a nadřízená jednotka opakuje vysílání po $T_{master-timeout}$.

3. Chybný počáteční nebo koncový znak, chybný kontrolní součet. Podřízená jednotka neodpoví, vyprázdní buffer a po době $T_{slave-timeout}$ je opět připravena na příjem zprávy. Nadřízená jednotka nedostane odpověď a po době $T_{master-timeout}$ příkaz zopakuje. Protože $T_{master-timeout} > T_{slave-timeout}$, zastihne nová zpráva podřízenou jednotku již opět připravenou přijímat.

4. Špatně přenesený délkový bajt nebo ztracená nebo přidaná data. Pokud přijde méně dat než se očekává podle hlavičky, podřízená jednotka po čekání na zbytek dat po době $T_{slave-timeout}$ vyprázdní vstupní buffer a připraví se na opakování zprávy, nadřízená jednotka opakuje příkaz stejně jako v bodě 3. Pokud přichází více dat, nesedí kontrolní součet ani koncový znak. Jednotka pak vyprazdňuje buffer dokud chodí přebytečná data a po době $T_{slave-timeout}$ je opět připravena na opakování příkazu.

Při přenosu odpovědi z podřízené jednotky nadřízené může dojít ke stejným situacím. V případě, že nadřízená jednotka přijme poškozenou zprávu, zahodí ji a opakuje příkaz nebo zahlásí chybu.

5.2.4 Příkazy

Příkazy pro komunikaci mezi CKDM a PC lze rozdělit do čtyř skupin.

1. Obecné příkazy přímo pro CKDM, ladicí příkazy

2. Příkazy pro rameno

3. Příkazy pro stůl

4. Příkazy pro luxmetr

Nebudu zde popisovat všechny jednotlivé příkazy, je jich více než třicet. Datový rámec vždy začíná řetězcem určujícím konkrétní příkaz, např. "T_Goto" je příkaz pro přesun stolu do zadané polohy. Následují parametry specifické pro daný přikaz. Odpověd na příkaz vždy začíná chybovým kódem, který určuje zda se vykonání příkazu zdařilo, nebo nastala nějaká chyba. Následovat mohou další data odpovědi, např. souřadnice stolu při dotazu na jeho polohu.

Příkazy a funkce pro komunikaci se všemi zařízeními musí být asynchronní, aby bylo možné ovládat více přístrojů najednou a ještě obsluhovat uživatelské rozhraní. Pokud tedy chci např. z PC přesunout stůl do určité polohy, učiním tak zasláním `T_Goto`. Ten v CKDM způsobí zavolání funkce `t_goto()`, která pošle stolu příkaz, aby se přesunul do zvolené polohy. Ihned poté se `t_goto()` ukončí. Po zavolání funkce CKDM pošle zpět PC odpověd na příkaz, v odpovědi je obsažena informace o tom, zda se příkaz pro započetí přesunu stolu zdařil. PC se potom periodicky příkazem `T_Flags` dotazuje, zda se stůl stále pohybuje. Když zjistí, že pohyb skončil, dotáže se příkazem `T_LastErr`, zda pohyb skončil úspěchem nebo nastala nějaká chyba při přesunu stolu. Tento princip je použitý i pro ovládání ramene a luxmetru.

5.3 Komunikace po sběrnici IEEE 488.1

Sběrnice GPIB [2] je průmyslová paralelní sběrnice určená především pro propojování digitálních měřicích přístrojů. Na sběrnici lze připojit až patnáct přístrojů, kabelová vzdálenost mezi dvěma přístroji nesmí přesáhnout dvacet metrů, maximální přenosová rychlost je 1 Mb/s.[3] Sběrnice používá třístavovou logiku, každý přístroj může vysílat buď logickou 1 (nízké napětí na sběrnici), logickou 0 (vysoké napětí na sběrnici), nebo může příslušný výstup uvést do stavu vysoké impedance. Tabulka 1 ukazuje výsledek logického součinu při dvou výstupech připojených na jeden vodič.

logická hodnota A	logická hodnota B	logická hodnota A.B
0	0	0
0	1	1
1	1	1
0	Hi-Z	0
1	Hi-Z	1
Hi-Z	Hi-Z	nedefinováno

Tabulka 1: Tabulka logického součinu na jednom vodiči sběrnice GPIB

Komunikace probíhá s pomocí následujících signálů:

- osm datových signálů DIO1 .. DIO8
- tři signály pro potvrzování předání každého bajtu: Data Valid (DAV), Not Ready For Data (NRFD), Not Data Accepted (NDAC)

[2] IEEE Std 488.1, někde se uvádí jako HPIB
[3] Novější verze umí pracovat i s vyšší přenosovou rychlostí

- pět řídících signálů pro různé účely:
  - Attention (ATN) rozlišuje mezi obyčejným datovým přenosem a příkazy sběrnice.
  - Interface Clear (IFC) umožňuje nastavit některé funkce rozhraní do definovaného stavu.
  - Service Request (SRQ) umožňuje připojeným zařízením požádat o přerušení přenosu a obsluhu výjimečné situace.
  - Remote Enable (REN) slouží k zakázání ručního ovládání vybraných přístrojů.
  - End or Identify (EOI) ukončuje přenos vícebajtové zprávy nebo vynucuje zahájení funkce Paralell Poll. Ta slouží k rychlému zjištění toho, zda některý přístroj nepotřebuje obsloužit výjimečnou situaci.

Komunikace po GPIB je implementována v souborech `gpib.c, gpib.h`.

Implementaci usnadnil fakt, že pouze jednotka CKDM je po celou dobu controllerem sběrnice. Vodiče sběrnice jsou připojeny k GPIO pinům procesoru přes budiče 75160 a 75161. Protože jednotka CKDM musí číst i zapisovat signály DAV, NRFD, NDAC, a nemusí vždy platit že výstupní úroveň na pinu procesoru je stejná jako úroveň na sběrnici[4], jsou tyto signály každý připojen na dva piny procesoru, jednou pro čtení a jednou pro zápis

Při navrhování funkcí pro komunikaci jsem se inspiroval funkcemi karty AX5488 z původního goniofotometru, dokumentovanými v [AX5488]. Modul `gpib` nabízí následující funkce:

- `gpib_init()`. Tato funkce provede inicializaci komunikačního modulu.
- `gpib_reset()`. Pokud selhala některá z ostatních funkcí, touto funkcí lze znovu inicializovat modul a pokusit se o nový přenos.
- `gpib_get_status()`. Pokud od posledního `gpib_reset()` nebo `gpib_init()` došlo k nějaké chybě, `gpib_get_status()` vrátí kód indikující jaká chyba nastala. Příkladem chyb je timeout, přerušení uživatelem nebo přetečení datového bufferu.
- `gpib_ifc()`. Tato funkce provede GPIB funkci Interface Clear, která uvede funkce SH a AH všech zařízení připojených na sběrnici do definovaného stavu.
- `gpib_gtl()`. Tato funkce povolí ruční ovládání daného přístroje (příkaz Go To Local).
- `gpib_get()`. Tato funkce pro daný přístroj vykoná GPIB funkci Group Execute Trigger.
- `gpib_mim()`. Tato funkce pošle danému přístroji specifikovanou GPIB zprávu Multiline Interface Message. Pro používané MIM kódy jsou k dispozici předdefinovaná makra.
- `gpib_read()`. Tato funkce přečte data od daného přístroje.
- `gpib_wrstr()`. Tato funkce pošle zadaný řetězec danému přístroji.

[4] viz tabulka 1

Implementace nízkoúrovňové komunikace po sběrnici GPIB si vyžádala mnoho týdnů ladění a studia [GPIB]. Sběrnice je přivedena na GPIO piny procesoru pouze přes budiče 75160, 75161. Implementoval jsem následující funkce:

- `SH1` (Source Handshake, complete capability)
- `AH1` (Acceptor Handshake, complete capability)
- `T8` (Talker, basic talker, no serial poll, not talk only mode, unaddress if MLA)
- `L4` (Listener, basic listener, not listen only mode, unadress if MTA)
- `C1, 2, 3, 28` (Controller, system controller, send IFC and take charge, send REN, not respond to SRQ, send interface messages, not receive control, not pass control, not pass control to self, not parallel poll, not take control synchronously.

5.4 Ovládání luxmetru

Komunikace s luxmetrem je implementována v souborech `luxmetr.c, luxmetr.h`.

Luxmetr má velmi jednoduché ovládání. Po přijetí zprávy `GET` [5] sejme aktuální hodnotu z displeje a uloží ji do bufferu. Poté lze z luxmetru vyčíst obsah bufferu funkcí `gpib_read()`. Luxmetr má sedm měřicích rozsahů, vybrat jeden z nich lze vysláním řetězce "`R0`" až "`R6`".

Při vyčítání hodnoty luxmetru může osvětlenost kolísat, proto se vyčítá opakovaně a hodnoty se průměrují. Navíc se v klouzavém okénku kontroluje, zda se od sebe naměřené hodnoty příliš neliší. Pokud je rozdíl naměřených hodnot příliš velký, pokračuje se v dalších odměrech dokud se osvětlenost neustálí. Při přepínání rozsahů je implementována hystereze, takže se nemůže stát, že by osvětlenost kolísající těsně na rozhraní nižšího a vyššího rozsahu způsobovala neustálé přepínání rozsahu.

5.5 Ovládání otočného stolu

Komunikace s otočným stolem je implementována v souborech `table.c, table.h`.

Otočný stůl má implementováno mnoho vlastních vysokoúrovňových příkazů, které jsou zdokumentovány v [stul]. Pomocí těchto příkazů lze přímo nastavit stůl do dané polohy nebo pohnout jím relativně k současné poloze, změnit nulovou polohu, provést zreferencování, zjistit kód poslední chyby, povolit nebo zakázat ruční ovládání z panelu a další.

Po započetí vykonávání příkazu stůl nereaguje na další příkazy až do ukončení provádění. Pokud je během provádění příkazu vyslán stolu další řetězec, stůl přijme první znak prvního řetězce, ale už nepřijme další znaky. Pro testování připravenosti stolu je vyslán prázdný příkaz " " a poté se testuje, zda stůl příkaz přijal nebo došlo k vypršení času vymezenému pro vyslání znaku.

[5] Group Execute Trigger

5.6 Ovládání ramene

Komunikace s ramenem je implementována v souborech `arm.c, arm.h`.

Krokový motor je poháněn jednotkou CD30x firmy MICROCON s.r.o. Ta obsahuje řadič krokových motorů M1486 téže firmy, výkonový stupeň, konektory pro připojení univerzálních vstupů/výstupů a konektor pro připojení kabelu rozhraní RS232. Použita je verze řadiče bez EEPROM, povelový soubor není nutné udržovat v paměti řadiče i po vypnutí napájení. Verze s EEPROM je také výrazně pomalejší v provádění příkazů zaslaných po sériovém rozhraní, protože povely se ukládají do trvalé paměti a po každém příkazu je nutné čekat 0,5 s než se příkaz do EEPROM uloží. Řadič nabízí mnoho užitečných funkcí:

- mikrokrokování
- různé tvary křivky náběhu a brždění motoru, nastavitelná start/stop rychlost, maximální rychlost i zrychlení
- univerzální vstupy a výstupy
- vstup Limit pro okamžité zastavení pohybu ramene s brzděním
- povelový soubor zahrnující více než 50 povelů
- komunikace po rozhraní RS232 rychlostí až 9600 baudů.

Bohužel, až po zakoupení řadiče a několika dnech programování jsem zjistil, že řadič má i své zápory. Setkal jsem se především s následujícími nedostatky:

- Dokumentace [M1486] není vždy úplně jednoznačná a úplná, v několika místech si lze vyložit popisovanou věc několika různými způsoby, u něčeho jsem nepochopil přesný princip a musel jsem postupovat metodou pokus-omyl.
- Prakticky nelze zjistit, zda se motor po zadání příkazu pro pohyb již zastavil. Řadič při provádění pohybu reaguje pouze na příkazy Kill a Reset. Lze sice za příkaz pro provádění pohybu zařadit příkaz Upload, který poté co pohyb skončí vyšle stavový bajt, ale pokud jednotka CKDM čeká na bajt který nepřichází, nikdy nemůže vědět zda nepřichází protože pohyb se stále ještě vykonává, nebo proto že došlo k přerušení spojení. V tomto případě lze pouze spoléhat na to, že spojení mezi CKDM a CD30x je funkční.
- Řadič neumožňuje nastavení záporné polohy, jako nulový úhel je proto definována polovina rozsahu kroků, tedy 8 000 000 nejmenších mikrokroků.
- Řadič neumožňuje zjistit aktuální polohu během provádění pohybu. Pokud tede začne pohyb ramene, jednotka CKDM si zaznamená čas jeho počátku a aktuální hodnotu zhruba počítá podle uplynulého času, nastavené rychlosti a nastaveného zrychlení.

- Pro komunikaci s řadičem jsem použil mód, kdy řadič opakuje všechny přijaté příkazy zpět na sériový port. Tento mód umožňuje kontrolovat, zda byl příkaz opravdu přijat. Bohužel, u příkazu Kill řadič ozvěnu neposílá, tento případ se tedy musí ošetřovat speciálně.

Všechny uvedené problémy se nakonec podařilo překonat, i když ne vždy úplně „čistě". Jsou implementovány všechny potřebné příkazy, tedy pohyb do dané polohy, zjištění polohy, nastavení nulové polohy, referencování ramene, spuštění nekonečného pohybu jedním nebo druhým směrem, zastavení pohybu. Je implementováno zastavení na koncových spínačích i zreferencování ramene. [6]

5.7 Uživatelské rozhraní

Uživatelské rozhraní je spolu s funkcemi pro komunikaci s PC implementováno v souborech `ui.c, ui.h`.

Otočný stůl má od začátku panel pro ruční ovládání přístroje, na který je obsluha zvyklá. Snažil jsem se tedy udělat i uživatelské rozhraní pro ruční ovládání ramene co možná nejpodobnější ovládání panelu stolu. Uživatelské rozhraní, implementované v souborech `ui.c, ui.h`, umožňuje následující:

- Posun po 0.05° oběma směry při opakovaném tisknutí pohybových kláves.
- Rychloposun při držení pohybových kláves.
- Zreferencování ramene.
- Nastavení nové nulové polohy ramene.

[6] Povolená oblast pohybu ramene je omezená indukčními koncovými spínači. Přejetí ramene za vymezené krajní polohy tyto spínače sepne, což má za následek okamžité zastavení ramene a nastavení příznaku „rameno není referencováno". Nereferencované rameno nemá definovanou polohu a před měřením je nutné jej nejprve zreferencovat.

Referencování provádí speciální povelový soubor. Cílem je přesunout rameno do polohy, ve které začíná spínat koncový spínač v kladném směru a prohlásit o tomto místě, že má určitou definovanou polohu. Ukáži zde celý okomentovaný povelový soubor, je z něj vidět i způsob programování řadiče.

```
  "[", /* začátek povelového souboru */
  "C7", /* zapnutí výkonového stupně */
  "T42", /* vypnutí kontroly vstupu Limit */
  ")14", /* pohyb mimo oblast sepnutí koncového spínače v záporném směru (vstup 14) */
  "C42", /* zapnutí kontroly vstupu Limit */
  "G+", /* kladný pohyb, ukončen při sepnutí vstupu Limit koncovým snímačem v kladném směru
*/
  "R", /* spuštění pohybu */
  "C52", /* zastavování na mikrokrocich, nikoli celokrocich */
  "T42", /* vypnutí vstupu Limit */
  "S128", /* nastavení rychlosti pohybu */
  "(15", /* pohyb v záporném směru, nalezení setupné hrany koncového spínače (vstup 15) */
  "C42", /* povolení vstupu Limit */
  "T7", /* vypnutí výkonového stupně */
  "=8000000", /* nastavení aktuální polohy na 8 000 000 */
  "U16", /* vyslání stavového bajtu nadřízené jednotce -- informace o ukončení referencování
*/
  "]", /* konec povelového souboru */
```

- Najetí do nulové polohy.
- Nastavení až čtyř uživatelských programů, tedy posunů danou rychlostí o daný úhel. Programování se provádí v dialogu vyvolaném stiskem kláves `Shift-F1` až `Shift-F4`.
- Vyvolání uživatelských programů stiskem kláves `F1` až `F4`.

Z displeje je využit první řádek na zobrazování aktuální polohy ramene a dodatečných informací. Pokud není rameno zreferencováno, zobrazuje se na prvním řádku symbol `?`. Pokud se čeká na dokončení pohybu ramene, zobrazuje se symbol hodin. Pokud je ovládání uzamčené, tj. rameno je ovládáno z PC, zobrazuje se na prvním řádku symbol `PC`. Další řádky jsou využity pouze při programování a pro zobrazování chyb, např. chyba komunikace s PC nebo pokus o ovládání vypnutého ramene.

5.8 Podpůrné moduly

Moduly budu uvádět v abecedním pořadí.

5.8.1 ad.c, ad.h

Velmi jednoduchý modul, slouží pouze k vyčítání A/D převodníku. Na vstupu A/D bude připojen potenciometr otáčející se zároveň s ramenem, jeho odpor tedy bude závislý na poloze ramene. Podle hodnoty odporu bude možné hrubě určit, zda se rameno nachází v požadované poloze, nebo se z nějakého důvodu (překážka, ztráta vazby motoru apod.) přestalo pohybovat.

Modul je hotový, ale polohu ramena zatím nekontroluji. Rameno není fyzicky nainstalované, hodnoty odporu potenciometru budu cejchovat až při oživování závěrečné instalace.

5.8.2 atol.c, atol.h

Modul obsahuje pouze jednu funkci, `atol()`, která převádí string na celé číslo.

5.8.3 block.c, block.h

Implementace některých užitečných funkcí pro práci s bloky paměti, které nejsou ve standardní knihovně, např. `memmove(), strcasecmp()` a další.

5.8.4 buzzer.c, buzzer.h

V jednotce CKDM je zabudován malý reproduktor – bzučák. Jeho vstup je přiveden na jeden z PPG [7] kanálů, takže programováním tohoto kanálu lze posílat na bzučák obdélníkový signál s různou frekvencí, střídou a počtem period. Modul `buzzer.c` poskytuje funkce pro inicializaci bzučáku, přehrání tónu o dané frekvenci a délce. Dále je zde implementováno několik funkcí pro přehrání standardních zvuků– zvuk OK, zvuk Chyba, zvuk Stisk klávesy.

[7] Programmable Pulse Generator

5.8.5 cmos.c, cmos.h

Tento modul se stará o ukládání a načítání proměnných které je potřeba uchovávat i když je CKDM vypnuté do/z paměti CMOS. Ukládají se všechny uživatelem nastavené programy pro rameno, na konec se uloží i jednoduchý kontrolní součet. Pokud při čtení součet neodpovídá, do globálních proměnných se uloží tovární hodnoty.

5.8.6 globals.h

Hlavičkový soubor společný všem modulům. Jsou zde makra určující název a verzi projektu, multiplatformní definice některých specifických typů (byte, uint16 apod.), často používaná makra (MIN, MAX, ABS) a další užitečné věci.

5.8.7 intvec.c

Zde jsou pomocí direktiv `#pragma` nastaveny priority a obslužné rutiny jednotlivých přerušení procesoru.

5.8.8 main.c

Modul obsahuje především funkci `main()`. Ta není složitá, nejprve inicializuje periferie a poté zavolá funkci `ui()` z modulu `ui.c`, která v nekonečné smyčce obsluhuje veškeré události.

5.8.9 misc.c, misc.h

Zde jsou různé užitečné funkce, které je lze jen obtížně zařadit do nějaké konkrétnější kategorie. Modul obsahuje funkce pro inicializaci veškerých portů, tisk ladicích zpráv na displej a další.

5.8.10 mutex.h

V tomto hlavičkovém souboru, jsou definována makra pro práci s jednoduchými mutexy pomocí vkládaného assembleru. Mutexy jsou typu spinlock, ve smyčce se neustále testuje hodnota mutexové proměnné.

5.8.11 num2str.c, num2str.h

V tomto modulu jsou definovány funkce pro převod čísla na string. V `num2str.h` je pak definováno mnoho užitečných maker pro převod v dvojkové, desítkové nebo šestnáctkové soustavě, na displej nebo do stringu.

5.8.12 p_num_x_x.c, p_num_x_x.h

Tento modul obsahuje funkce pro práci s čísly s posunutou desetinnou čárkou. Tato čísla jsou reprezentována jako typ int nebo long int, ale při zobrazování nebo načítání se tváří jako reálná čísla s desetinnou čárkou na pevné pozici. Například číslo `12345` s čárkou posunutou o tři řády (posunutí je jedním z parametrů funkcí z `p_num_x_x`) se zobrazí jako `12.345`. Použití těchto

čísel může být v mnoha případech výhodnější než použití typu float, zvláště pokud není potřeba ukládat rozsah hodnot přes několik řádů. Nedochází ke ztrátě přesnosti, mají menší paměťové nároky, práce s nimi je jednodušší a rychlejší.

5.8.13 panel.c, panel.h

Modul obsluhuje klávesnici, displej, podsvícení displeje a LED diody jednotky CKDM.

5.8.14 parse_float.c, parse_float.h

Modul převádí string s reálným číslem na typ float. Umí obyčejný zápis i zápis v tzv. vědecké notaci (např. `11.3E-5`), kladná i záporná čísla. Součástí jsou i jednotkové testy převodní funkce. Převodní funkce byla speciálně navržena tak, aby dokázala korektně zpracovat všechny formáty čísel, které může stůl po sběrnici GPIB poslat.

5.8.15 rct.c, rtc.h

Modul zprostředkovává komunikaci s čipem reálných hodin (RTC). Obsahuje funkce pro nastavování i čtení data a času a komunikaci s pamětí CMOS.

5.8.16 sleep.c, sleep.h

Modul umožňuje využít funkcí procesoru pro nastavování speciálních módů šetřících spotřebu.

5.8.17 timer.c, timer.h

Modul umožňuje měřit čas s rozlišením až 10 ms. Obsahuje i funkci pro čekání po určitou dobu. Aby bylo snazší některé funkce ladit mimo jednotku CKDM, je psán tak, aby šel přeložit pro CKDM i pro OS Windows. Pod CKDM využívá přerušení od PPG kanálu 1, v obslužné rutině tohoto přerušení mimo jiné inkrementuje časový čítač `jiffies`. Pod Windows modul využívá funkci `GetSystemTime()`.

5.8.18 uart.c, uart.h

Modul realizuje nízkoúrovňovou komunikaci přes rozhraní RS232. Je psán tak, aby šel přeložit pod OS Windows i přímo pro CKDM. Má funkce pro asynchronní i synchronní čtení a zápis, realizuje vlastní bufferování příchozích dat. Definicí příslušných maker a opětovným překladem kódu lze volit, zda se bude či nebude bufferování používat.

6 Software

Obrázek 13: Úvodní okno aplikace Goniofotometr

6.1 Úvod

Návrh a vývoj software byla nejdelší[1], ale také nejzajímavější částí celé diplomové práce. V této kapitole popíši požadavky kladené na plikaci Goniofotometr, důvody které mě vedly ke zvolení jazyka Python, architekturu celého software a poté podrobněji rozeberu důležitější moduly. Aplikaci Goniofotometr budu v této kapitole nazývat pouze goniofotometr, pokud to nepovede k nejasnostem.

6.2 Požadavky na aplikaci

6.2.1 Definice typových zkoušek

Aplikace musí umět odměřit charakteristiky daného zdroje světla a vyhodnotit, zda vyhovuje daným požadavkům. Tento proces se nazývá **typová zkouška**. Aplikace musí mít rozhraní pro definici typových zkoušek pro všechny druhy návěstidel a svítidel, tj. pro směrová návěstidla, sestupová návěstidla, všesměrová návěstidla a svítidla. Speciální třídou typových zkoušek jsou tabulkové typové zkoušky, u kterých je definována minimální svítivost přímo pro vybrané body. Slouží především k rychlému ověření správné funkčnosti návěstidla v kritických bodech.

6.2.2 Odměr dat

Aplikace musí mít uživatelské rozhraní k měření dat pro vyhodnocení typových zkoušek, pro odměry dat individuálního vývoje, pro měření kalibrační matice a pro další potřeby. Měřit musí být možné bez přípravku, s přípravkem nebo s pomocí ramene. Během měření musí být zřejmé, jaká část již byla odměřena. V některých případech je vhodné zobrazovat průběžný náhled již odměřených dat.

6.2.3 Vyhodnocování typových zkoušek

Naměřená data musí být možné vyhodnotit podle zvolené definice typové zkoušky. Vyhodnocení se liší podle typu návěstidla, v některých případech zahrnuje izokandelový diagram, v někte-

[1] Vývoj software trval zhruba 800 hodin čistého času, výsledkem je téměř 20 000 řádků zdrojového kódu.

rých případech polární grafy. Musí být možné již provedené vyhodnocení uložit a později jej znovu otevřít. Vyhodnotit lze buď již dříve naměřená data, nebo podle definice typové zkoušky naměřit nová a vyhodnotit ta. Při vyhodnocení typů měření, u kterých to dává smysl, musí být možnost použít tzv. simulovaný filtr a/nebo kalibrační matici. [2]

6.2.4 Individuální vývoj

Elektrosignál vyvíjí i nové modely svítidel a návěstidel. Při vývoji hrají velkou roli zkušenost a cit, i velmi malá změna polohy vlákna žárovky nebo napájecího proudu se může projevit významnou, i když pro laika jen těžko předvídatelnou změnou vyzařovací charakteristiky světelného zdroje. Jeden z požadavků na aplikaci je poskytnout nástroje pro usnadnění takového vývoje.

6.2.5 Kalibrace stolu, ramene a luxmetru

Nulové úhly stolu a ramene musí být přesně nastaveny, jinak by goniofotometr měřil zdroje světla v chybné poloze. Jedním z požadavků na software je vytvoření uživatelského rozhraní pro nastavení nulové polohy stolu a ramene a pro nastavení pozic snímačů luxmetru.

6.2.6 Ukládání dat

XML[3] je rozšířený, jednoduchý, textový, přenositelný jazyk, který popisuje hierarchická data. Jednotlivým uzlům stromu mohou být přiřazeny atributy dourčující aplikačně závislé vlastnosti.

V aplikaci Goniofotometr musí být ve formátu XML možné ukládat a načítat veškerá používaná data, tj. definice typových zkoušek, odměřená data, výsledky vyhodnocení zkoušek, rozpracované individuální vývoje, nastavení programu a další. Cílem je do budoucna umožnit jiným programům nebo databázím snadný import a export dat aplikace. Navíc musí být možné data měření ukládána ve formátu CSV[4], tedy jako řádky hodnot oddělených čárkami.

6.2.7 Export a tisk protokolů

Formát PDF[5] byl navržen firmou Adobe, je dobře specifikovaný, přenositelný a výborně se hodí pro tisk. Jednou z hlavních výhodou formátu je, že vzhled stránky je identický na monitoru i na výsledné tiskové straně.

Veškeré protokoly o měření a vyhodnocení musí být možné ukládat ve formátu PDF. Vytisknutý protokol musí být identický s dokumentem uloženým v PDF souboru.

6.2.8 Hlášení chyb

Program musí srozumitelně hlásit chyby, které nastanou během provádění akcí volených uživatelem. Pokud dojde k vnitřní chybě programu, musí být pokud možno i tato ohlášena uživateli

[2] viz kapitola 6.10
[3] eXtensible Markup Language
[4] Comma Separated Values
[5] Portable Document Format

a zaprotokolována. Tyto protokoly mohou později sloužit jako podklady při opravě nově zjištěných chyb.

6.2.9 Nastavení programu

S pomocí uživatelského rozhraní musí být dostupná konfigurace programu, tedy cesta k programu Adobe Reader, volba používaného sériového portu, nastavení rozlišení obrázků a další.

6.3 Volba programovacího jazyka

6.3.1 Porovnání některých jazyků

V dnešní době existují stovky programovacích jazyků, z nich několik desítek je aktivně používáno. Výběr správného jazyka pro větší softwarový projekt může velmi významně ovlivnit dobu vývoje, hardwarové nároky i výslednou kvalitu projektu. Kritérií je přitom mnoho a často jsou vzájemně protichůdná.

Při výběru jazyka jsem nejprve vyloučil všechny, které jsou speciálně navrženy tak, aby bylo obtížné v nich programovat[6]. Poté jsem vyloučil jazyky pro funkcionální a logické programování, protože s nimi nemám příliš zkušeností a nechtěl jsem osud projektu příliš ponechávat náhodě. Vlastnosti rozšířenějších nebo zajímavějších jazyků, jsem shrnul v tabulce 2. Splnění každé vlastnosti je ohodnoceno jedním až pěti body, čím více bodů, tím lepší výsledek. Tabulka je samozřejmě pouze orientační, není podložena ověřenými údaji, kromě toho pro každý projekt jsou důležité jiné vlastnosti.

Vlastnost/Jazyk	C#	C++	D	Delphi	Java	Python	Ruby	Smalltalk
Rozšířenost	3	5	1	3	4	3	2	1
Knihovny	4	5	1	4	5	4	2	1
Jednoduchost	2	1	2	4	2	4	4	5
Rychlost vývoje	3	2	3	3	3	4	4	5
Hustota zápisu	2	1	2	2	2	5	5	3
Komunita	4	4	1	4	5	4	3	2
Podpora OOP	3	2	3	3	3	4	4	5
Nároky výsledné aplikace	2	4	4	3	2	1	1	3
Součet	23	24	17	26	26	29	25	25

Tabulka 2: Orientační porovnání programovacích jazyků

Nakonec jsem zvolil jazyk Python, především pro jeho jednoduchost, úsporný a přesto dobře čitelný styl programování, velké množství dostupných knihoven a výbornou podporu objektově

[6]Takovým jazykem je například Malbolge, jazyk pojmenovaný po osmém pekelném kruhu Inferna Dante Aligieriho. Chování každé instrukce závisí na její pozici v paměti modulo 94, všechny instrukce jsou samomodifikující, ukazatele na datovou a programovou oblast se každý takt virtuálního stroje inkrementují. Pouhé tři registry všechny pracují v trojkové soustavě. Není divu, že první program v Malbolge se objevil až po dvou letech od vzniku jazyka, kromě toho nebyl napsán člověkem, ale vygenerován jiným počítačovým programem. Program vygenerovaný heuristickým prohledáváním stavového prostoru všech programů v Malbolge dokázal vypsat na standardní výstup řetězec „HEllO WORld". Dnes je situace o něco lepší, existuje program který dokáže kopírovat standardní vstup na standardní výstup a je popsán postup, jak napsat program pro tisk libovolného řetězce. I přesto je však programování v Malbolge výrazně složitější než třeba v jazyce C++.

orientovaného programování. Svoji roli hrály také osobní sympatie, již před započetím práce na aplikaci Goniofotometr jsem v Pythonu napsal několik kratších programů a vždy jsem byl s tímto jazykem spokojen.

Samořejmě žádný jazyk není dokonalý, a ani Python v tomto ohledu není výjimkou. Asi nejvíce jsem se potýkal s téměř naprostou absencí statické kontroly. Python je jazyk s pozdní vazbou jmen na objekty, překlep například v názvu proměnné se tedy projeví až při provádění daného příkazu vyvoláním výjimky. Těmto problémům by však bylo možné z velké části předcházet při použití vývojového prostředí, které by upozorňovalo na nedeklarované proměnné. Vyzkoušel jsem několik vývojových prostředí, ale žádné mi plně nevyhovovalo, nakonec jsem program psal v textovém editoru Vim. Je ovšem pravdou, že je velmi obtížné provádět kontrolu jmen objektů v jazyce, kde například příkaz

```
setattr(o, 'hel'+'lo', 'world')
```

vytvoří objektu `o` novou instanční proměnnou `hello` s řetězcovou hodnotou `'world'`. Existující nástroje pro statickou analýzu kódu[7] dokáží na některé chyby upozornit, ale mnohá varování vygenerovaná těmito nástroji mohou být falešná.

Jazyk Python je také v porovnání s jinými jazyky poměrně pomalý a náročný na pamět. Uvádí se, že programy v Pythonu běží asi dvacetkrát pomaleji než ekvivalentní programy v jazyce C, ale velmi záleží na konkrétní situaci. Na druhou stranu je program v Pythonu napsán mnohem dříve, takže programátorovi zbyde více času na optimalizace. Pokud je to nutné, lze časově kritické části kódu přepsat např. do jazyka C a poté je volat z pythonovského programu.

6.4 Použité externí knihovny

Jazyk Python má v základní instalaci přibaleno množstvím užitečných knihoven všeho druhu, samozřejmě ale není možné umístit do instalace všechny existující knihovny a snažit se tak pokrýt požadavky všech uživatelů. Aplikace Goniofotometr vyžaduje ke svému běhu některé další knihovny[8], použity jsou ovšem pouze knihovny se svobodnou licencí.

- **wxPython** je vazba knihovny wxWidgets[9] pro jazyk Python. wxPython slouží k tvorbě sofistikovaných uživatelských rozhraní, obsahuje velké množství tříd a funkcí pokrývajících 99% všech myslitelných požadavků na uživatelské rozhraní. Knihovna má dlouhou historii, je promyšleně navržena, výborně zdokumentována a jednoduše se používá. Navíc je multiplatformní, pracuje pod většinou unixových systémů, dvaatřicetibitovými Windows a pod Macintosh OS X. Na každé platformě využívá mateřské ovládací prvky, tedy pod Windows používá WinAPI, pod unixovými systémy využívá knihovnu GTK, pod OS X spolupracuje s rozhraním Aqua.
  Knihovna wxPython je dostupná na <`http://www.wxpython.org`>

[7] Příkladem takového nástroje je `pychecker`, viz URL <`http://pychecker.sourceforge.net`>
[8] Instalační balíčky knihoven jsou součástí přiloženého CD.
[9] Knihovna wxWidgets je psána v jazyce C++, ale existují vazby na mnoho jiných jazyků. Knihovna je dostupná na <`http://www.wxwidgets.org`>.

- **Matplotlib** je sofistikovaná knihovna pro tvorbu 2D grafů. Mottem tvůrců je: *„Matplotlib tries to make easy things easy and hard things possible"*. Knihovna nabízí velké množství nejrůznějších druhů grafů – čárové, koláčové, chybové, histogramy, dvourozměrné plochy vyjádřené barvou a další. Pro renderování používá knihovnu AntiGrain[10], takže výsledné grafy vypadají opravdu velmi dobře. Vykreslení grafu s vyhodnocením typové zkoušky trvá zhruba půl vteřiny na PIII 1 Ghz.
  Knihovna Matplotlib je dostupná na <`http://matplotlib.sourceforge.net`>.

- **NumPy** je knihovna pro rychlé vědecké výpočty. Umožňuje práci s n-rozměrnými maticemi s volitelnou přesností uložených čísel, nabízí mnoho užitečných matematických operací a funkcí, Fourierovu transformaci a další.
  Knihovna NumPy je dostupná na <`http://numpy.scipy.org`>

- **ReportLab** je jednoduchá, ale mocná knihovna pro tvorbu PDF dokumentů. Knihovna umožňuje pracovat s různými řezy i barvami písem, dokáže automaticky zalamovat text, vkládat obrázky, dokonce umí zpracovat podmnožinu jazyka HTML[11].
  Knihovna ReportLab je dostupná na <`http://www.reportlab.org/rl_toolkit.html`>

- **pyRXP** je velmi rychlá knihovna pro načítání XML souborů. Snadno se používá, načtený XML soubor převádí na strom realizovaný proměnnými typu tuple.
  Knihovna pyRXP je dostupná na <`http://www.reportlab.org/pyrxp.html`>

- **pyWin32** umožňuje pod OS Windows využívat funkce WinAPI[12]. PyWin32 je nutnou prerekvizitou pro instalaci knihovny pySerial pod OS Windows.
  Knihovna pyWin32 je dostupná na <`http://sourceforge.net/projects/pywin32`>

- **pySerial** je multiplatformní knihovna pro přístup k sériovému portu (rozhraní RS232). Nutnou prerekvizitou pro používání pySerial pod OS Windows je instalace knihovny pyWin32.
  Knihovna pySerial je dostupná na <`http://pyserial.sourceforge.net`>

- **amoeba** je krátká knihovna implementující simplexovou metodu pro hledání maxima funkce jedné či více proměnných.
  Knihovnu lze stáhnout z <`http://stitchpanorama.sourceforge.net/Python/amoeba.py`>.

6.5 Architektura aplikace

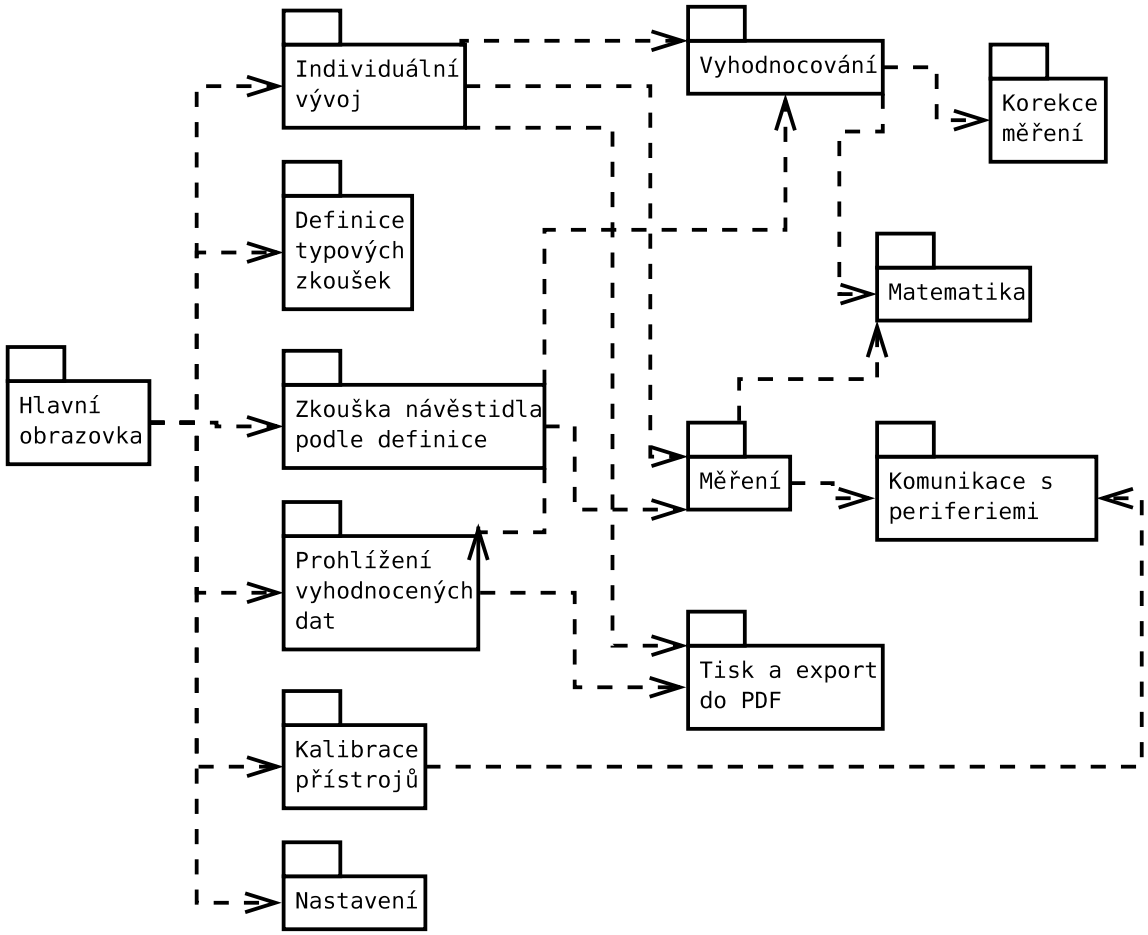
Aplikace goniofotometr je středně rozsáhlý projekt, a samozřejmě bylo nutné ji rozčlenit na několik dílčích modulů s pevně vymezenými funkcemi. Snažil jsem se navrhnout každý modul pokud možno samostaný a soběstačný. Několik modulů je ovšem samozřejmě natolik integrální

[10] AntiGrain je knihovna grafických algoritmů zaměřená na vysoký výkon a nejvyšší možnou kvalitu. Domovská stránka je na <`http://www.antigrain.com/`>.

[11] Hypertext Markup Language, v tomto jazyce jsou psány WWW stránky.

[12] Windows Application Programming Interface. Rozhraní, které umožňuje volat funkce operačního systému Windows.

součástí celé aplikace, že na nich závisí téměř všechny ostatní. Rozdělení aplikace na jednotlivé části je zobrazeno na schématu 14.



Obrázek 14: Rozdělení aplikace Goniofotometr na jednotlivé moduly

Téměř téměř všechny části aplikace závisí na modulu `mvc`, který implementuje návrhový vzor Model View Controller, blíže popsaný v kapitole 6.6. Ty části programu, které nějakým způsobem pracují s naměřenými daty, navíc závisí na modulu `storage,` který implementuje datové kontejnery a algoritmy pro naměřená data. Pokud aplikace potřebuje kreslit grafy, využívá modul `plot.` Tyto závislosti ve schématu nejsou zakresleny, příliš mnoho vazeb mezi moduly v jednom obrázku by srozumitelnosti spíše ublížilo. Na schématu nejsou zakresleny ani méně významné moduly, například modul `bicubic` implementující bikubickou interpolaci, nebo modul `amoeba` pro hledání extrému funkce více proměnných.

6.6 Implementace vzoru Model View Controller

Jedním ze základních pravidel objektového programování je striktní rozdělení zodpovědností jednotlivých objektů. Každý objekt by měl být pokud možno co nejvíce samostatný a specializovaný na právě jednu úlohu. Pokud spolupracuje s jinými objekty, měl by pracovat pouze s nimi a nikoliv s třetími objekty získanými voláním metod těchto objektů. Tato idea byla částečně formalizována např. v [Demeter88].

Pro oddělení logiky aplikace a jejího uživatelského rozhraní se často používá návrhový vzor **Model View Controller**, dále jen **MVC**. Ten jsem použil i já, a jeho implementace tvoří ústřední část návrhu celé aplikace.

Princip je následující:

- **Model** reprezentuje data, se kterými aplikace pracuje. Příkladem modelů použitých v aplikaci jsou: odměřená data, poloha ohniska, datum měření, graf, barva a mnoho dalších. Model většinou obsahuje nějaká data, existují ale i bezestavové modely. Model není zodpovědný za svou grafickou reprezentaci, ani za to, zda je právě zobrazován.

- **View** je pohled na konkrétní instanci modelu. Příkladem je seznam položek, textový vstupní řádek nebo zaškrtávací políčko. Pohledy mohou být i složené, například pohled na model „příprava na měření" zobrazuje mnoho položek pro definici typu měření, nastavení přístrojů a textových popisků. Vnitřně každý pohled obsahuje referenci na model, který zobrazuje. Jeden model nemusí být zobrazován žádným pohledem, ale může být také zobrazován mnoha různými pohledy. Například model barvy může být zároveň zobrazován posuvníky pro červenou, zelenou a modrou složku a zároveň barevným trojúhelníkem. Pokud se hodnoty barvy změní, dojde k automatické aktualizaci příslušných pohledů.

- **Controller** reaguje na události od uživatele a spojuje události v pohledech s odezvou v příslušných modelech, má však nejméně jednotnou definici. Moje implementace controlleru reaguje na události systému wxWidgets, jako jsou například aktivace položky v seznamu, nebo změna hodnoty vstupního políčka. Controller obsahuje reference na příslušný pohled i model.

Je nutné nějak propojit události provedené v modelu (například změnu hodnoty modelu celého čísla) s aktualizací příslušného pohledu. Goniofotometr používá podobný mechanismus jako ostatní implementace MVC.

1. Každý model implementuje metodu `add_change_observer(func)`. Tato metoda zařadí volatelný objekt func do svého seznamu pozorovatelů.

2. Každý model implementuje metodu `changed()`. Ta zavolá všechny pozorovatele ze seznamu.

3. Každý pohled obsahuje referenci na právě jeden model. Každý pohled implementuje metodu `update()`. Tato metoda má za úkol vykreslit nebo překreslit pohled na referencovaný model.

4. Při inicializaci každého pohledu si tento zaregistruje u svého modelu svoji metodu `update` voláním `model.add_change_observer(update)`. Od této chvíle bude informován o případných změnách stavu svého modelu.

5. Pokud model změní svůj stav, např. voláním metody `substract()` u modelu celého čísla, zavolá metodu `changed()`. Tím informuje všechny pozorovatele o tom, že se jeho stav změnil a bude tedy třeba překreslit případné pohledy.

6. Pokud je zavřen pohled na nějaký model, odregistruje svoji funkci `update` ze seznamu pozorovatelů voláním metody `del_change_observer()` svého modelu.

Hlavní výhodou tohoto přístupu je oddělení aplikační logiky od prezentace, tedy separace zodpovědností jednotlivých objektů.

Pro goniofotometr jsem implementoval vlastní verzi MVC. Uvažoval jsem o použití již hotových knihoven, ale nenašel jsem žádnou, která by splňovala všechny mé požadavky a byla dostatečně dokumentovaná a stabilní. Nejslibněji vypadal projekt Traits a TraitsUI firmy Enthought[13], ale ještě nedávno byl k dispozici pouze pro Python verze 2.3, zatímco goniofotometr je implementovaný v Pythonu verze 2.4. Dnes již je k dispozici i verze Traits pro Python 2.4, myslím si že je to v současnosti nejlepší implementace MVC frameworku pro jazyk Python.

Moje implementace MVC pro goniofotometr má následující vlastnosti:

- Implementuje základní modely pro všechny často používané datové struktury: bezestavového společného předka `Model`, `String` (řetězec), `Float` (reálné číslo), `Int` (celé číslo), `Bool` (logická hodnota pravda/nepravda), `StrEnum` (výčet řetězcových možností), `Filename` (název souboru), `Version` (verze), `Composite` (model obsahující libovolné další modely), `List` (uspořádaný seznam modelů explicitně vyjmenovaných tříd), `Array` (uspořádaný seznam modelů jediné třídy), `FloatContainer` (zabaluje několik proměnných typu reálné číslo), `Point` (bod), `Rect` (obdélník definovaný dvěma rohy), `Color` (barva), `NumpyArray` (pole z knihovny NumPy), `Time` (datum a čas).

- Implementuje často používané pohledy: `View` (společný předek všech pohledů), `StaticText` (neměnný řetězec), `StaticTextBool` (textová reprezentace modelu `Bool`), `Text` (vstupní řádka pro libovolnou hodnotu), `Str` (vstupní řádka pro řetězec), `Int` (vstupní řádka pro celé číslo), `Float` (vstupní řádka pro reálné číslo), `Check` (zaškrtávací políčka), `Radio` (výběr z několika možností), `Gauge` (procentuální ukazatel), `List` (seznam, modelem je `List`), `Array` (seznam, modelem je `Array`), `Color` (zobrazení barvy), HTML (zobrazí řetězec jako HTML).

- Pro všechny pohledy, u kterých to má smysl, jsou implementovány příslušné controllery.

- Implementovány jsou tzv. editory, které zapouzdřují funkcionalitu příslušného pohledu a controlleru v jedné třídě. Například třída StrEditor je potomkem třídy StrView, ale ve svém konstruktoru vytvoří i instanci třídy StrController a automaticky ji propojí s příslušným pohledem, tedy sama se sebou.

- Modelu lze dočasně zakázat informování svých pozorovatelů metodou `freeze_changes()`. Zasílání upozornění se opět povolí metodou `thaw_changes()`.

- Každý model může implementovat metodu `validate()`, která má za úkol kontrolovat, zda jsou data modelu platná. Přetížením metody `validate()` lze například omezit platný

[13] Domovská stránka Traits a TraitsUI je na <`http://code.enthought.com/traits/`>

rozsah hodnot pro model úhlu od $-180°$ do $180°$. Controllery pak mohou použít metodu `validate()` k ověření toho, že uživatel v dialogu vyplnil platná data.

- Každý model má implementovány metody pro serializaci a deserializaci do/z souborů ve formátu XML. Protože většina nových modelů vzniká skládáním několika jiných modelů s pomocí `Composite` modelů, i tyto nové modely se umí automaticky ukládat a načítat ve formátu XML. Při načítání se kontroluje, zda je XML soubor korektní, chyby jsou automaticky hlášeny uživateli. Tímto mi odpadlo mnoho starostí s psaním ukládacích a načítacích metod pro každý objekt zvlášť.

- Každý model má metodu `copy()` pro vytvoření své kopie, a metodu `copy_from()` pro nastavení svých hodnot podle hodnoty jiného modelu stejného typu. Tyto jsou v podstatě analogiemi kopírovacího konstruktoru a přetíženého operátoru přiřazení z jazyka C++.

- Každý model má informaci o tom, zda byl od posledního načtení nebo uložení změněn. Tato informace výrazně usnadňuje rozhodování, kdy se ptát uživatele při zavírání okna s rozpracovaným dokumentem, zda jej chce uložit, nebo provedené změny zahodit.

Modul `mvc` nabízí i další užitečné třídy a funkce, zde je pouze stručně popíši:

- Funkce `log_model_errors()` informuje uživatele o chybě při kontrole příslušného modelu. Metoda `report_validation_problem` hledá chybu v hierarchii modelů, přičemž se snaží najít takový model, který je v této hierarchii nejhlouběji a zároveň je pro něj otevřený nějaký pohled. Tato metoda nalezne příslušný pohled a zavolá jeho metodu `notify_validation_error()`, která způsobí vizuální indikaci místa chyby[14]. Metoda dále ohlásí uživateli chybu a přesune kurzor na chybný pohled. Díky tomuto mechanismu hlášení chyb je zcela oddělena kontrola modelů od jejich vizuální prezentace.

- Třída `FileDialogs` usnadňuje a sjednocuje práci s dialogy pro ukládání a načítání souborů.

- Třída `TopLevelWindow` reprezentuje okno na obrazovce. Obsahuje metody pro počítání aktuálně otevřených oken programu, otevírání a zavírání okna, kontrolu zobrazeného modelu. Z třídy `TopLevelWindow` jsou odvozeny třídy `Dialog` a `Frame,` které poskytují další možnosti jako je načítání menu, modální a nemodální zobrazení daného okna, kontrolu uživatelem vyplněných dat apod.

- Z Třídy `Frame` je odvozena třída `DocumentFrame`, která sjednocuje okna s otevřeným dokumentem. Automaticky poskytuje funkcionalitu položek menu Otevřít, Uložit, Uložit jako..., Zavřít.

- `TextEntryDialog` je jednoduchý dialog pro vstup řetězce.

[14] Nejčastěji změna barvy pozadí

V celém programu jsem se snažil důsledně oddělovat aplikační logiku a prezentaci uživateli. Soubory končící na příponu `_gui` obsahují pouze uživatelské rozhraní pro příslušné moduly s datovými kontejnery a algoritmy.

Implementace MVC je v souboru `mvc.py`. Podrobnější informace o MVC lze dohledat např. v [mvc].

6.7 Hlavní obrazovka

Hlavní obrazovka[15] sestává pouze z loga goniofotometru a z menu, ze kterého jsou přístupné všechny funkce goniofotometru. Implementace je v souboru `main.py`, který zároveň slouží jako hlavní spustitelný soubor celé aplikace. Hlavní okno je možné zavřít pouze pokud není otevřené žádné další okno.

6.8 Definice typových zkoušek

Definice typových zkoušek je implementována v souboru `typetests.py`, uživatelské rozhraní je potom v souboru `typetests_gui.py`. Typové zkoušky se dělí na tři základní typy.

6.8.1 Izokandelové typové zkoušky

Izokandelové typové zkoušky jsou takové, které se definují pomocí izokandelových křivek. Měření probíhá v mřížkovém rastru pro azimut a elevaci s konstantním krokem. Graf takové zkoušky obsahuje 2D plochu popisující svítivost v jednotlivých bodech mřížky pomocí barvy[16], a jednotlivé definované a naměřené izokandelové křivky a jejich popisky.

Mezi izokandelové typové zkoušky patří typové zkoušky směrových a sestupových návěstidel.

6.8.2 Polární typové zkoušky

Pro polární typové zkoušky je definována minimální svítivost pro celé rozsahy bodů v azimutu nebo v elevaci. Příkladem takové typové zkoušky je:

- Intenzita pro azimut 0° a (0° < elevace < 60°) musí být vyšší než 500 cd.
- Intenzita pro azimut 180° a (0° < elevace < 60°) musí být vyšší než 500 cd.
- Intenzita pro elevaci 15° a (−180° < azimut < 180°) musí být vyšší než 300 cd.
- Krok měření je jeden stupeň.

Grafem takové zkoušky jsou dva polární grafy. U jednoho grafu je vždy konstantní azimut, u druhého elevace. Úhel v polárním grafu reprezentuje nekonstantní veličina, vzdálenost od středu je svítivost.

[15] Viz obrázek 13

[16] Tuto plochu nazývám v goniofotometru luminogram.

6.8.3 Tabulkové typové zkoušky.

Tabulkové typové zkoušky jsou tvořeny seznamem bodů a minimální svítivosti, kterou zdroj světla musí v těchto bodech svítit, přičemž požadovaná svítivost se zadává pro každý bod zvlášť. Měří se přímo v požadovaných bodech. Tabulkové typové zkoušky nemají žádný graf.

6.8.4 Implementace definic typových zkoušek

Všechny definice typových zkoušek mají společného předka, `Typetest`. Z něj dědí třídy `TypetestIso`, `TypetestPolar`, `TypetestTable`. Z třídy izokandelových typových zkoušek dědí třídy `TypetestDirectional` a `TypetestApproach`, tedy třídy popisující směrová a sestupová návěstidla. Z třídy `TypetestPolar` dědí třídy `TypetestOmni` a `TypetestLights`, popisující všesměrová návěstidla a svítidla. Třída `TypetestTable` již žádné potomky nemá. Tato hierarchie a terminologie je použita i v případě vývoje nebo měření. Těchto pět základních typů měření se prolíná celou aplikací.

Požadované izokandelové křivky izokandelových typových zkoušek jsou reprezentovány třídou `Isocandel` a jejími potomky. Každý potomek popisuje jiný typ izokandely. Jsou implementovány `IsocandelRectangle`, `IsocandelEllipse` a `IsocandelOval`, přičemž první dvě izokandely se používají pouze u směrových návěstidel a poslední typ pouze u sestupových návěstidel.

Rozsahy pro definici polárních typových zkoušek jsou popsány instancemi tříd `OmegaSpan` a `SigmaSpan`, kde `OmegaSpan` popisuje rozsah azimutu pro konstantní elevaci a `SigmaSpan` popisuje rozsah elevace pro konstantní azimut. Každý rozsah definuje počáteční a koncový úhel, krok měření a minimální požadovanou intenzitu.

Tabulková typová zkouška obsahuje seznam instancí třídy `TablePoint`. Každá instance definuje azimut, elevaci a minimální požadovanou svítivost.

6.9 Měření

Základní třídy pro měření jsou implementovány v souborech `measure.py` a `measure_gui.py`. Většina aplikace využívá přímo funkce těchto modulů, některé části si ale odvozují vlastní potomky se speciálními vlastnostmi.

V modulu `measure` je definována třída `Measure`, která definuje body, které se mají odměřit. Veřejný protokol je jednoduchý.

- Třída se inicializuje do definovaného stavu metodou `__init__`. Potomci třídy mají další inicializační metody, které umožňují například nastavit měření podle definované typové zkoušky, pro danou mřížku nebo pro daný rozsah.

- Třída implementuje metodu `next_point()`. Ta vrací souřadnice dalšího bodu, který se má odměřit.

- Metoda `reset_points()` způsobí, že příští volání `next_point()` vrátí opět první bod měření.

Třídy modulu `measure` jsou dále využívány modulem `measure_gui`, který obsahuje třídy GUI pro přípravu k měření i pro měření samotné.

V dialogu „příprava k měření" uživatel zadá definici pracoviště, tj. polohu čidla luxmetru a typ měření (bez přípravku, s přípravkem nebo s ramenem). Během přípravy upevní na stůl goniofotometru měřený zdroj světla. V této fázi je možné pohybovat se stolem a orientovat tak zdroj světla do definované polohy pomocí natočení a naklopení upínací plochy stolu. Uživatel vyplní doplňující údaje, jako je popis měření a poznámka, a spustí samotný odměr. Instance třídy `MeasureFrame` pak provede samotný odměr nad modelem `Measure`. Naměřená data ukládá do instance třídy `MeasData` z modulu `storage`. V modulu `measure_gui` je implementováno několik potomků třídy `MeasureFrame`, kteří během měření vykreslují náhled dosud zobrazených dat. `MeasureFramePoints` zobrazuje odměřená data jako seznam bodů, `MeasureFrameIso` jako izokandelovou mřížku. `MeasureFrameIso` ovšem narozdíl od `MeasureFramePoints` vyžaduje, aby byly měřené body rozmístěny v pravidelném rastru, takže jej nelze využít např. pro tabulková měření.

Odměr probíhá ve třech vláknech. Jedno vlákno obsluhuje události GUI, druhé vlákno dopředu počítá souřadnice stolu pro dané pozorovací úhly, třetí vlákno se stará o komunikaci s jednotkou CKDM. Měření trvá velmi dlouho, jeden bod se odměřuje zhruba pět sekund, takže je důležité nezdržovat měření ničím nepotřebným. Vícevláknový přístup problém řeší.

6.10 Vyhodnocování typových zkoušek

Obecné algoritmy pro vyhodnocování jednotlivých požadavků na zdroje světla jsou implementovány v souboru `eval.py`. Konceptuálně je modul rozdělen na třídy, které slouží pouze k vyhodnocování, a třídy, které slouží pouze k ukládání naměřených dat. Vyhodnocovací třídy jsou potomky základní třídy `Eval`. Všechny vyhodnocovací třídy jsou bezestavové, tj. mají pouze třídní metody sloužící pro výpočty a testy. Výsledky vyhodnocení se ukládají do instancí potomků třídy `EvalResult`. Třída `EvalResult` a její potomci slouží jednak jako úložiště vyhodnocených dat, jednak mají implementované metody ke zjišťování, zda vyhodnocení proběhlo úspěšně, pro vykreslení výsledku vyhodnocení do grafu a pro vygenerování HTML protokolu.

Modul `eval_typetest` potom z těchto tříd dědí do vlastních tříd `EvalTypetest` a `EvalTypetestResult`. Ty jsou již přímo použitelné pro vyhodnocování konkrétních typových zkoušek a ukládání výsledků vyhodnocení typových zkoušek.

Důvod pro toto dělení je takový, že algoritmy z modulu `eval` používá i modul individuálního vývoje `devel`. Ten však potřebuje pouze tyto algoritmy a nemá žádnou zodpovědnost za typové zkoušky, takže bylo nezbytné celé vyhodnocování rozdělit.

Výsledek vyhodnocení typových zkoušek je možné prohlížet v oknech implementovaných v souboru `eval_typetest_gui`. Tato okna jsou poměrně jednoduché pohledy, obsahují žádný, jeden nebo dva grafy a HTML protokol o vyhodnocení. Z menu těchto oken je možné exportovat a tisknout výsledné protokoly.

Při vyhodnocování je možné uplatnit tzv. simulovaný filtr a kalibrační matici.

Simulovaný filtr slouží pro měření návěstidel, která se budou používat s různými barev-

nými filtry. Není ovšem nutné ověřovat, zda návěstidlo vyhoví pro všechny barvy filtru. Pokud mají filtry známou propustnost, stačí naměřená data vynásobit touto propustností a vyhodnotit výsledná data.

Kalibrační matice slouží pro měření svítidel, jejichž vyzařovací charakteristika je závislá na jejich orientaci. Pro svítidlo je možné odměřit vyzařovací charakteristiku v referenční poloze, a poté v nějaké jiné poloze. Podílem naměřených rastrů získáme kalibrační matici. Později lze měřit svítidlo v poloze pro kterou byla odměřena kalibrační matice a vynásobit naměřená data touto maticí. Výsledek by měl být podobný měření přímo v referenční poloze.

Obě korekce jsou implementovány v souboru `meas_corrections.py`. Třídy `SimFilter` a `CalibrationMatrix` jsou určeny k používání zprostředkovaně přes třídu `MeasCorrector`.

6.11 Individuální vývoj

Jedním z požadavků na aplikaci bylo vytvoření nástrojů pro usnadnění vývoje nových svítidel a návěstidel. Pro tento vývoj používám termín **individuální vývoj.** Ten je realizován v souborech `devel.py` a `devel_gui.py` zvlášť pro tři případy. Pro každý případ lze tisknout a exportovat protokoly, rozdělanou práci je možné uložit a vrátit se k ní později.

6.11.1 Vývoj izokandelových návěstidel

Tento vývoj se týká směrových a setupových návěstidel, z popisovaných tří vývojů je zdaleka nejzajímavější. Okno je rozděleno na dvě části. V levé části je izokandelový graf, v pravé části je jakýsi „poznámkový blok". Do grafu lze přidávat nové definované izokandelové křivky a vyhodnocovat je, možné je i vyhodnotit izokandelu na datech jiného měření a zobrazit ji v aktuálním vývoji. To může být výhodné např. pro srovnání několika různých úprav návěstidla. Levým tlačítkem myši je možné izokandely přesouvat, pravým tlačítkem myši je možné izokandely mazat, nebo si nechat vyhodnotit extrémy svítivosti, střední svítivost a další. Výsledky vyhodnocení se automaticky píší do poznámkového bloku na pozici kurzoru, ovšem je možné si v bloku psát i vlastní poznámky. Již naměřenou mřížku lze přiměřit při libovolném okraji.

6.11.2 Vývoj všesměrových návěstidel a svítidel

Vývoj vypadá velmi podobně jako vyhodnocení, ale je možné dodatečně doměřovat a vyhodnocovat další rozsahy.

6.11.3 Vývoj tabulkových měření.

Vývoj vypadá velmi podobně jako vyhodnocení, ale je možné dodatečně doměřovat a vyhodnocovat další body tabulky.

6.12 Ukládání naměřených dat

V souboru `storage.py` jsou datové kontejnery pro naměřená data. Zásadní jsou tři třídy:

`MeasInfo` popisuje informace o měření, tj. orientaci a polohu zdroje světla, polohu snímače luxmetru, datum měření, poznámky k měření a další.

`MeasData` obsahuje přímo informace o odměřených bodech. Pro každý bod jsou uloženy souřadnice natočení a naklopení stolu, naměřená osvětlenost, ale také požadované a přepočítané hodnoty pozorovacích úhlů a svítivost. Cílem této redundance je mít možnost přepočítat souřadnice znovu v případě chybného vyplnění údajů o měření.

`Mesh` je kontejner pro ukládání bodů v mřížce s pevným rastrem. Pokud jsou `MeasData` přibližně v mřížce, dokáže je `Mesh` převést na skutečnou mřížku interpolací.

`MeasData` a `MeasInfo` obsahují metody pro export informací ve formátech ASCII a HTML.

6.13 Tisk a export protokolů

Většina protokolů je vnitřně v goniofotometru psána v malé podmnožině formátu HTML. Modul `pdf.py` dokáže generovat PDF dokumenty z této podmnožiny HTML, z textu ve formátu ASCII nebo z obrázků na disku.

Tisk probíhá tak, že se nejprve požadovaný tiskový materiál převede do formátu PDF, uloží se na disk do dočasného souboru, a poté se spustí program Adobe Reader s parametrem `/p` a jménem tohoto dočasného souboru. Tím je uživateli nabídnut standardní dialog pro tisk programu Adobe Reader.

6.14 Komunikace s přístroji

Komunikace je realizovaná v souborech `ckdm_interface.py`, `ckdm_interface_common.py`, `ckdm_interface_real.py`, `ckdm_interface_simul.py`. Jsou implementovány dvě verze:

Simulovaná verze nepotřebuje ke svému běhu skutečné přístroje, před měřením se pouze zeptá uživatele na vzorec, podle kterého se mají počítat simulované hodnoty osvětlenosti. Tato verze je implementována v souboru `ckdm_interface_simul.py`.

Skutečná verze komunikuje s přístroji prostřednictvím jednotky CKDM přes sériový port. Princip komunikace je vysvětlen v kapitole 5.2. Tato verze je implementována v souboru `ckdm_interface_real.py`.

Vybrat používanou verzi lze zakomentováním nebo odkomentováním příslušného řádku v souboru `ckdm_interface.py`.

6.15 Převod mezi polohou stolu a pozorovacím úhlem

Vzájemný převod mezi polohou stolu a pozorovacím úhlem je implementován v souboru `table_math.py`. Podrobný popis převodu je vysvětlen v kapitole 4.

6.16 Práce s grafy

Práce s grafy implementována v souboru `plot.py` je plně realizována v duchu Model View Controller. Třída `PlotModel` popisuje zobrazené čáry, popisky, mřížku a další prvky grafu. Třída

`PlotView` pak realizuje pohled na daný model grafu a třída `PlotController` umožňuje interakci s uživatelem. Reakce na uživatelovy akce jsou nastavitelné pomocí mechanizmu zpětných volání, callbacků. Graf zobrazený instancí třídy `PlotView` je možné exportovat na disk jako soubor s obrázkem, čehož je využito při tvorbě PDF protokolů.

6.17 Ošetření chyb

Chyby lze v zásadě rozdělit na takové, které způsobil uživatel a na chyby způsobené chybou v programu

6.17.1 Chyby způsobené uživatelem

Příkladem chyby způsobené uživatelem je pokus o vyhodnocení izokandelové typové zkoušky na datech, která nepokrývají celou oblast, která se má vyhodnotit.

Při ošetřování chyb jsem hojně používal techniky zvané řetězení výjimek[17]. Pokud se budu držet příkladu se špatným vyhodnocením zkoušky, pak metoda `_test_isocandel` třídy `EvaluatedDefIsocanel` vyvolá výjimku `EvalError` při zjištění, že izokandela přesahuje oblast daných naměřených dat. Tuto výjimku zachytí metoda `eval_typetest` a vyvolá další výjimku, `EvalTypetestError`. Tuto výjimku nakonec zachytí funkce `gui_ask_eval_iso` z modulu `eval_typetest_gui`.

wxPython obsahuje užitečnou funkci, `wx.LogError()`. Při každém zavolání se přidá argument – textový popis chyby – do vnitřního seznamu chyb. Při další iteraci hlavní smyčky wxPython zjistí, zda je seznam prázdný. Pokud ano, nic nedělá. Pokud seznam obsahuje jednu chybu, wxPython zobrazí dialog s touto chybou. Pokud seznam obsahuje více, chyb, wxPython zobrazí dialog s poslední chybou v seznamu. Dialog v tomto případě ovšem obsahuje tlačítko `Více`, po jehož stisku se dialog rozbalí a uživatel si může přečíst celý seznam chyb, přičemž nejstarší je na prvním místě.

Tato funkce je v aplikaci hojně využívána. Opět se vrátím k příkladu se špatným vyhodnocením. Při prvním vyvoláním výjimky se zavolá např. `wx.LogError('Přesah na hranici sigma min.')` a hned poté `wx.LogEror('definovaná izokandela se nevejde do naměřených dat')`. Když výjimku `EvalError` zachytí metoda `eval_typetest`, zavolá `funkci wx.LogEror('Chyba při vyhodnocování typové zkoušky')`. Uživatel pak vidí chybové dialogové okno s textem 'Chyba při vyhodnocování typové zkoušky'. Pokud si okno rozbalí, může si přečíst že se izokandela nevejde do naměřených dat, pokud jej zajímá ještě více detailů, dozví se dokonce na jaké hranici izokandela data překračovala.

[17] Exception Chaining

6.17.2 Chyby programu

1. Každý program obsahuje alespoň jednu chybnou instrukci
2. Každý program lze alespoň o jednu instrukci zkrátit.
Z toho plyne: Každý program lze zkrátit na jedinou, a to chybnou instrukci.

Žádný program není bezchybný, tedy ani goniofotometr, i když bych si jistě přál opak. Pokud už však k chybě dojde, neměla by skončit pádem aplikace, ale slušným upozorněním na vnitřní chybu a zaprotokolováním této chyby, aby bylo později snadno možné ji opravit. V Pythonu lze takové chování naštěstí implementovat snadno. Moje implementace je v souboru `error_logging.py`.

Standardní modul `sys` obsahuje proměnnou `excepthook`, která referencuje obslužnou funkci nezachycených výjimek. Prostým přiřazením lze nastavit vlastní obsluhu těchto výjimek. V případě goniofotometru obslužná funkce spustí dialog s upozorněním na chybu. Do dialogu může uživatel vepsat informace, které se mu zdají důležité. Poté je do XML souboru chyba zaprotokolována, a to včetně jmen a hodnot všech lokálních proměnných a obsahu zásobníku virtuálního stroje.

7 Závěr

Práce shrnuje výsledky rok a půl trvajícího úsilí o modernizaci zkušebny svítidel a letištních návěstidel. V rámci práce byl vytvořen zcela nový software pro měření zdrojů světla a vyhodnocování získaných dat, dále byl navrhnut a implementován firmware průmyslového mikropočítače CKDM sloužícího k ovládání otočného ramene a jako rozhraní mezi měřicími přístroji. Zcela nově byl vyřešen převod mezi polohou stolu a pozorovacím úhlem měřeného zdroje světla. Jedna kapitola práce je věnována obecnému popisu problematiky letištních návěstidel a požadavků na ně kladených.

Vytvořený software je hotový a v testovacím užívání. Lze měřit skutečné zdroje světla s pomocí goniofotometru, nebo provádět simulovaná měření.

Firmware jednotky CKDM je hotový a odzkoušený, komunikace mezi přístroji je funkční, rameno lze ovládat vzdáleně i prostřednictvím ručního ovládání implementovaného v jednotce CKDM.

Otočné rameno zatím není zabetonováno do stěny zkušebny, takže nelze měřit svítidla ve vodorovné poloze. Aplikace Goniofotometr zatím pracuje v režimu simulovaných měření, není tedy nutné fyzicky propojovat počítač s přístroji. Až bude rameno instalováno do konečné polohy, počítač se připojí k jednotce CKDM přes rozhraní RS232 a aplikace bude přepnuta do režimu skutečných měření.

8 Seznam literatury

[Annex] ICAO. *Annexes to the Convention on International Civil Aviation* [CD-ROM]. 2006 Edition.
Lze objednat na URL: <http://www.icao.int>. ISBN 92-9194-647-8. Annex 14.

[zkratky] Air Navigation Services of the Czech Republic. *Zkratky používané v publikacích AIS* [online].
Dostupné na <http://ais.ans.cz/ais_data/aip/data/valid/g2-2.pdf>.

[Veselý97] Veselý, l. *Měření letištních návěstidel.* Praha, 1997.
Doktorská práce na Fakultě elektrotechnické Českého vysokého učení technického.

[rotace] Murray, G. *Rotation About an Arbitrary Axis in 3 Dimensions* [online].
Dostupné na <http://www.mines.edu/~gmurray/ArbitraryAxisRotation/ArbitraryAxisRotation.html>

[crc] Ross W. *A Painless Guide to CRC Error Detection Algorithms* [online].
Dostupné na <http://www.geocities.com/SiliconValley/Pines/8659/crc.htm>

[AX5488] AXIOM Technology. *AX5488 GPIB Interface Card User's Manual.*
Taiwan : AXIOM Technology, 1994

[GPIB] ANSI/IEEE. *Std 488IEEE Standard Digital Interface for Programmable Instrumentation.* Std 488.1-1987. The Institute of Electrical and Electronics Engineers, New York, 1988. ISBN 471-62222-2.

[M1486] MICROCON. *Kontroler pro krokový motor M1486 : Uživatelský manuál.*
Praha : MICROCON s.r.o.

[CD30x] MICROCON. *Deska pro řízení krokových motorů CD30x.*
Praha : MICROCON s.r.o.

[recipes] Press, W., Flannery, B., Teukolsky, S., Vetterling, W.
Numerical recipes in C: The art of scientific computing. 2^{nd} edition.
Cambridge University Press, 1992. ISBN 0-521-43108-5.

[stul] Štajnochr L. *ROTAČNÍ STŮL DVOUOSÝ : Návod k obsluze a programování systému.* Praha : MIKRONEX a.s., 1993.

[Demeter88] Karl J. L., Holland I., Arthur J. R. *Object-oriented programming: An objective sense of style* [online]. 1988 [cit 1. května 2006].
Dostupné na <http://www.ccs.neu.edu/research/demeter/papers/law-of-demeter/oopsla88-law-of-demeter.pdf>

[mvc] BURBECK, S. *Applications Programming in Smalltalk-80^{TM}: How to use Model-View-Controller (MVC)* [online]. 1992 [cit. 20. listopadu 2006]. Dostupné na <http://st-www.cs.uiuc.edu/users/smarch/st-docs/mvc.html>

[malbolge] SCHEFFER, L. *Introduction to Malbolge* [online]. 2005 [cit. 4. ledna 2006]. Dostupné na <http://www.lscheffer.com/malbolge.shtml>

Rejstřík

Annex 14, 2
azimut, 3

CD30x, 9
CKDM, 10

Elektrosignál, 1
elevace, 3
ELTODO Power s.r.o., 1

goniofotometr, 7
GPIB, sběrnice, 28
GPIO, 11

hlavní svazek, 4

ICAO, 2
IEEE 488.1, sběrnice, 28
individuální vývoj, 48
izokandelové diagramy, 4
izokandelové typové zkoušky, 45

kalibrační matice, 48
kandela, 7

lux, 7

model view controller, 42
MVC, 42

osvětlenost, 7

PAPI, 5
PDF, 37
polární typové zkoušky, 45

rameno, 10

simulovaný filtr, 47
svítivost, 7

tabulkové typové zkoušky, 46
typová zkouška, 36

VASIS, 4

XML, 37
XML, načítání a ukládání, 44

A Anglicko-český slovníček některých pojmů

Přidávám česko-anglický slovníček některých letištních pojmů používaných v [Annex]. Čerpal jsem především z [zkratky] a od zaměstnanců firmy Elektrosignál.

aerodrome (aeronautical) beacon: letištní maják
approach lightning system: přibližovací světelný systém
apron: odbavovací plocha
circling guidance lights: naváděcí světla pro let okruhem
crossbar: příčka
displaced treshold: předsunutý práh
docking: zavádění letadla na stání
elevated (lights): nadzemní (návěstidla)
guidance (lights): naváděcí (návěstidla)
holding: vyčkávání
identification beacon: poznávací maják
inset (lights): zapuštěná (návěstidla)
intensity (low/medium/high): intenzita (nízká/střední/vysoká)
non-directional: nesměrový
obstruction, obstacle: překážkový, překážka
omnidirectional: všesměrový
PAPI: viz **precision path approach indicator**
precision approach lightning system: přesný přibližovací systém kategorie I, II nebo III
precision path approach indicator: systém indikace sestupové přibližovací roviny
rapid exit taxiway indicator lights: návěstidla pro rychlý výjezd z dráhy
runway centre line lights: osové světelné řady RWY
runway edge lights: postranní dráhové řady
runway end lights: koncové světelné příčky RWY
runway lead-in lightning system: zaváděcí světelný systém RWY
runway touchdown zone lights: dotykové postranní řady
runway: vzletová a přistávací dráha
simple approach lightning system: jednoduchá přibližovací světelná soustava
stopway: dojezdová dráha
stopway lights: dojezdová světelná soustava
taxiway-link: spojovací dráha
taxiway: pojezdová dráha
touchdown zone: dotyková zóna
VASIS: viz **visual approach slope indicator system**
visual approach slope indicator system: světelná sestupová soustava
wing bar lights: vnější polopříčky

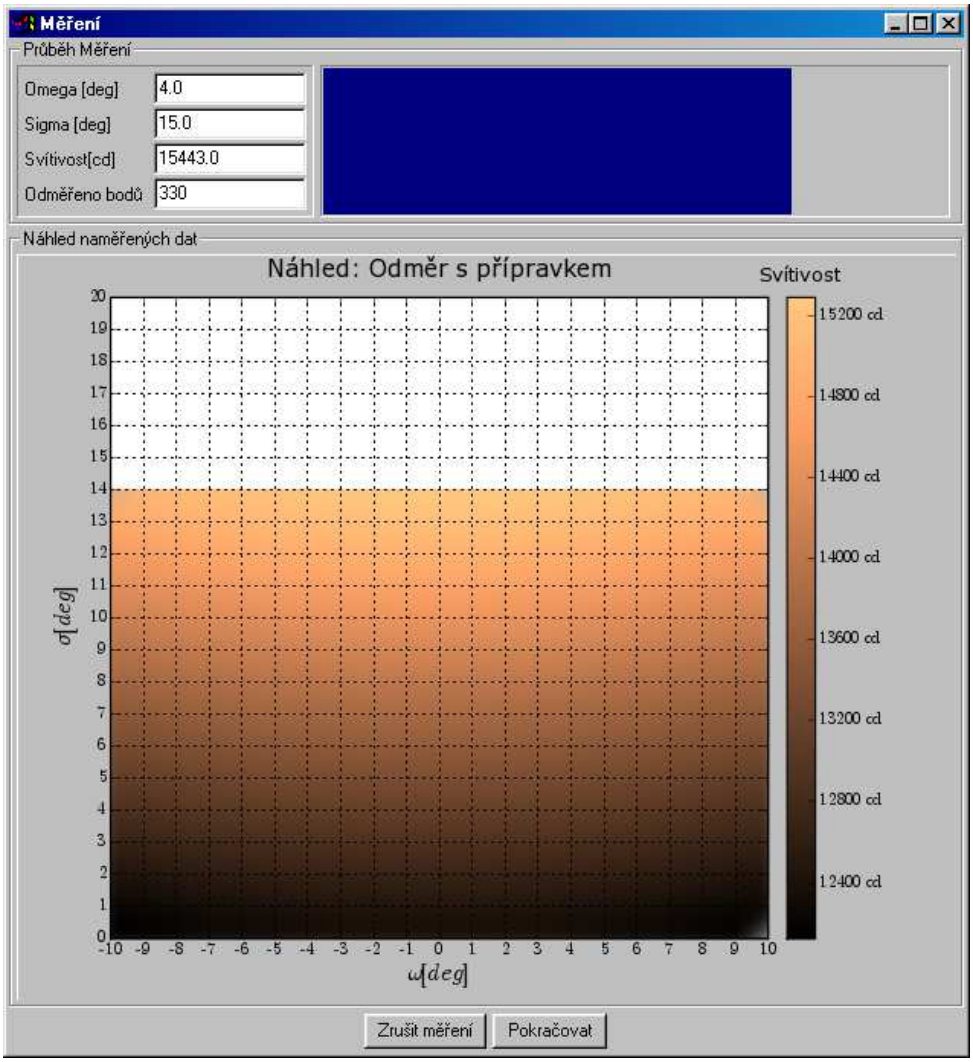
B Obsah přiloženého CD

Přiložené CD obsahuje následující soubory a adresáře:

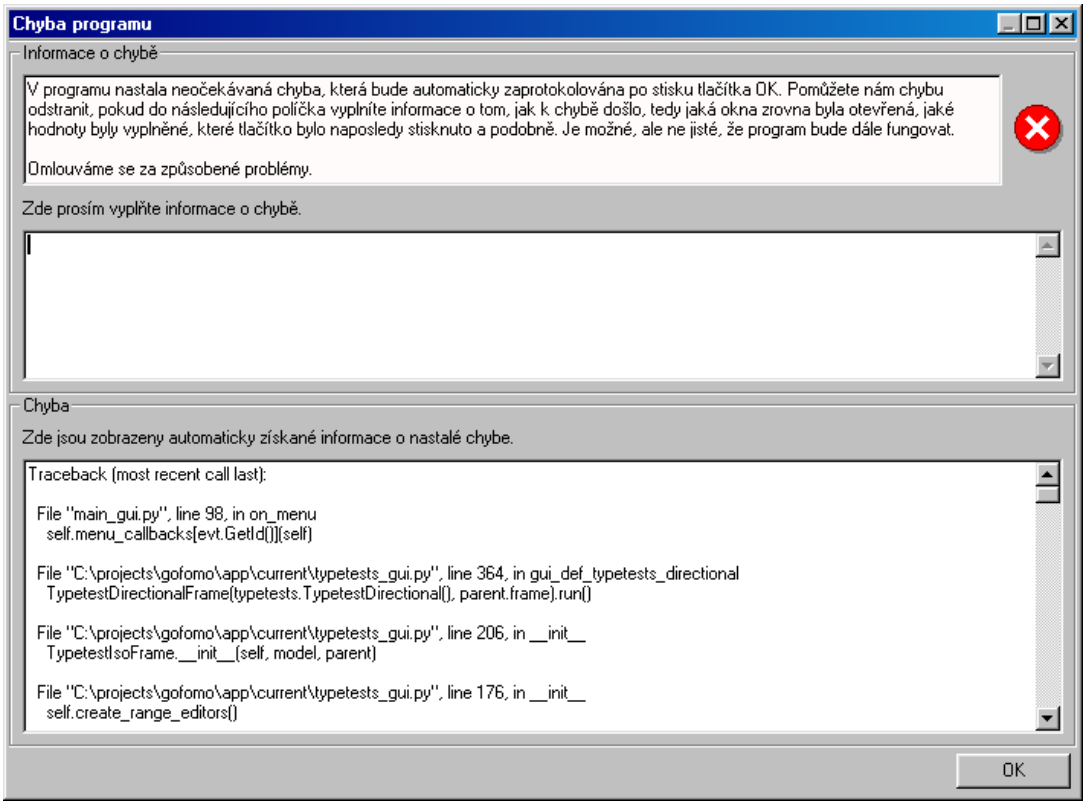
- **README** – Tento soubor popisuje obsah přiloženého CD. Všechny podadresáře kořenového adresáře CD obsahují vlastní soubory README, které se týkají přímo jich samotných.
- **ckdm** – V tomto adresáři jsou zdrojové kódy firmware jednotky CKDM.
- **dp** – V tomto adresáři jsou zdrojové kódy a PDF verze této diplomové práce.
- **gfm** – V tomto adresáři jsou zdrojové kódy aplikace Goniofotometr.
- **install** – V tomto adresáři jsou instalační balíčky Pythonu a knihoven potřebných ke spuštění aplikace Goniofotometr.

C Obrazové přílohy

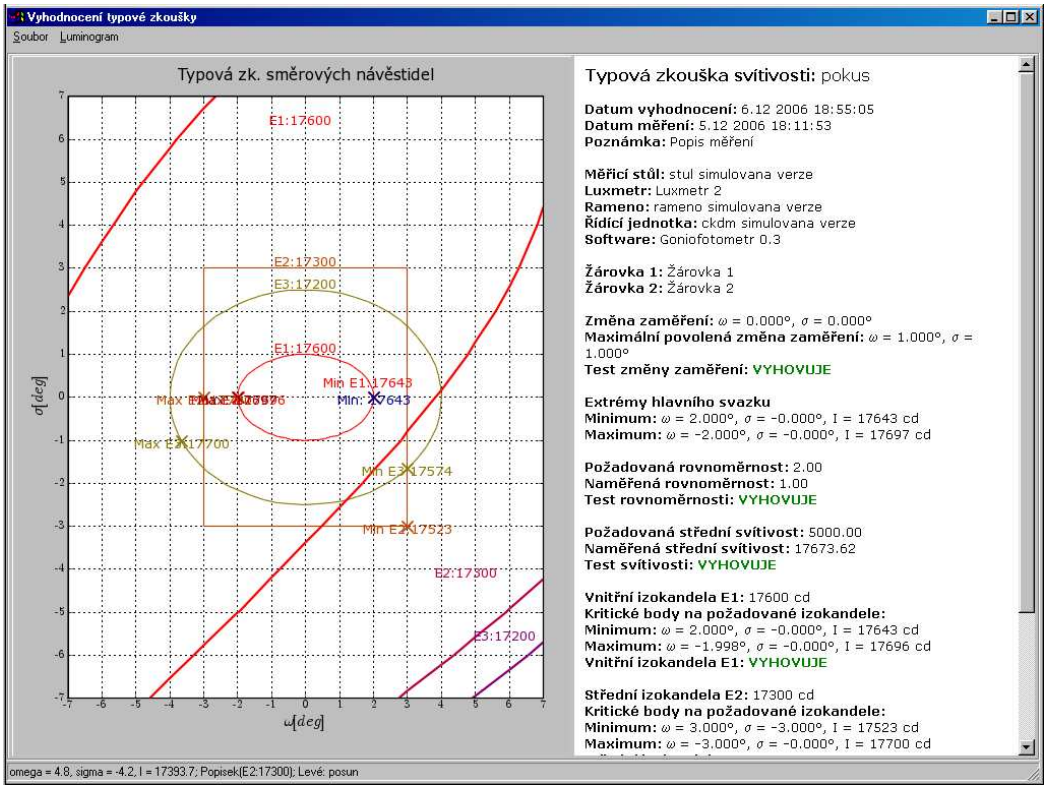
Tato příloha obsahuje obrázky aplikace Goniofotometr.



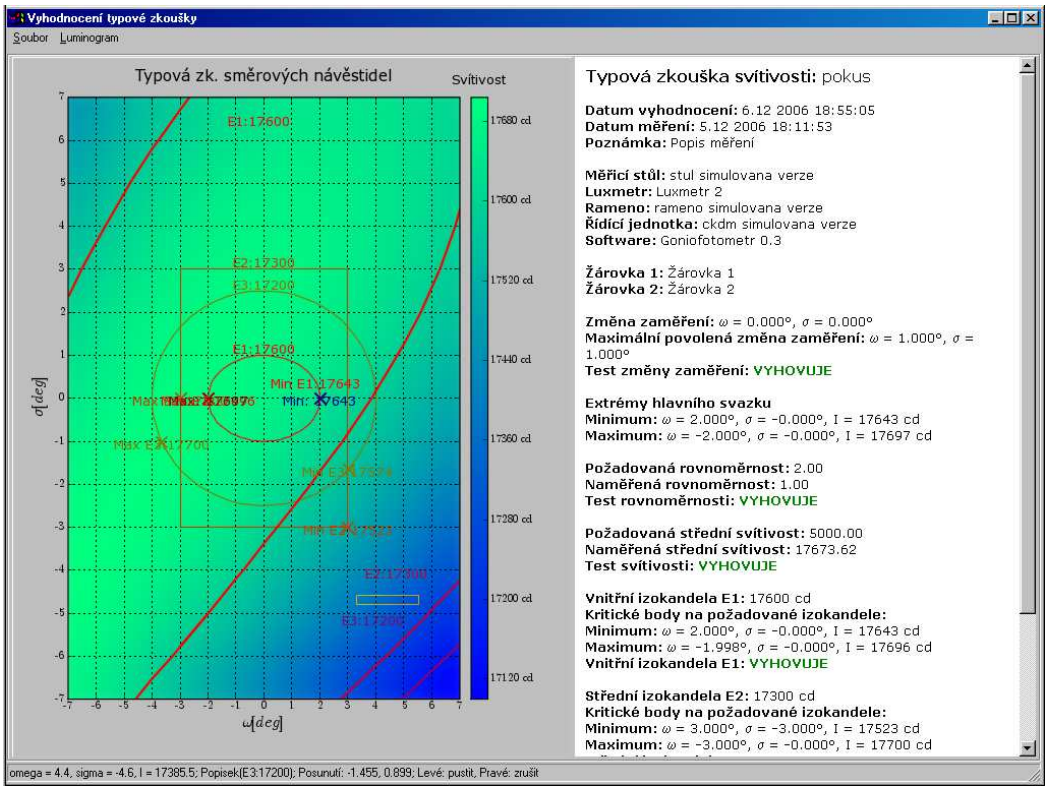
Obrázek 15: Odměřování kalibrační matice



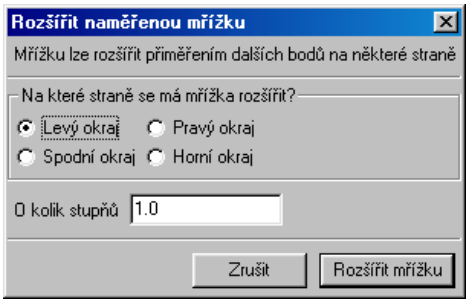
Obrázek 16: Hlášení chyby programu



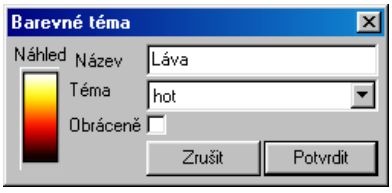
Obrázek 17: Vyhodnocení typové zkoušky, graf bez luminogramu



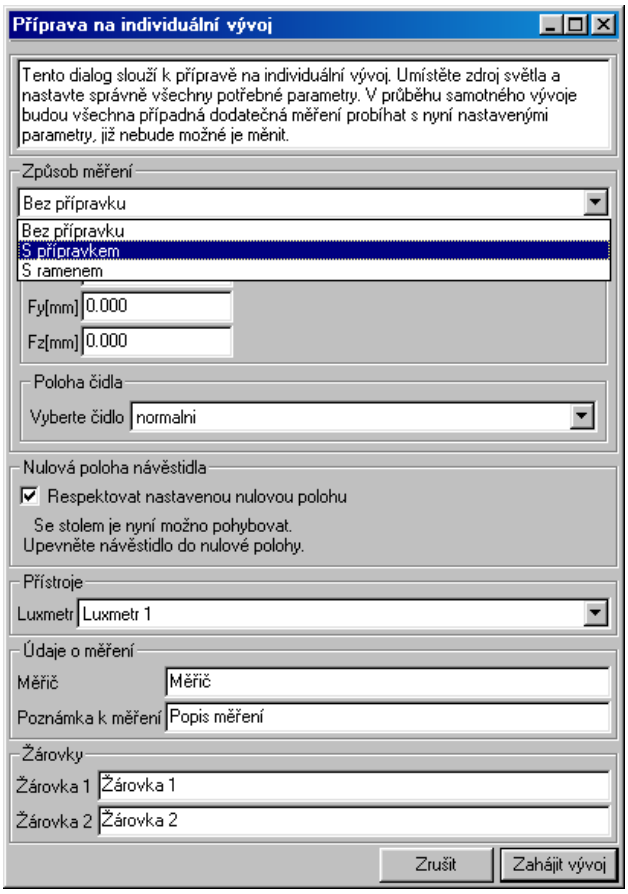
Obrázek 18: Vyhodnocení typové zkoušky, graf s luminogramem



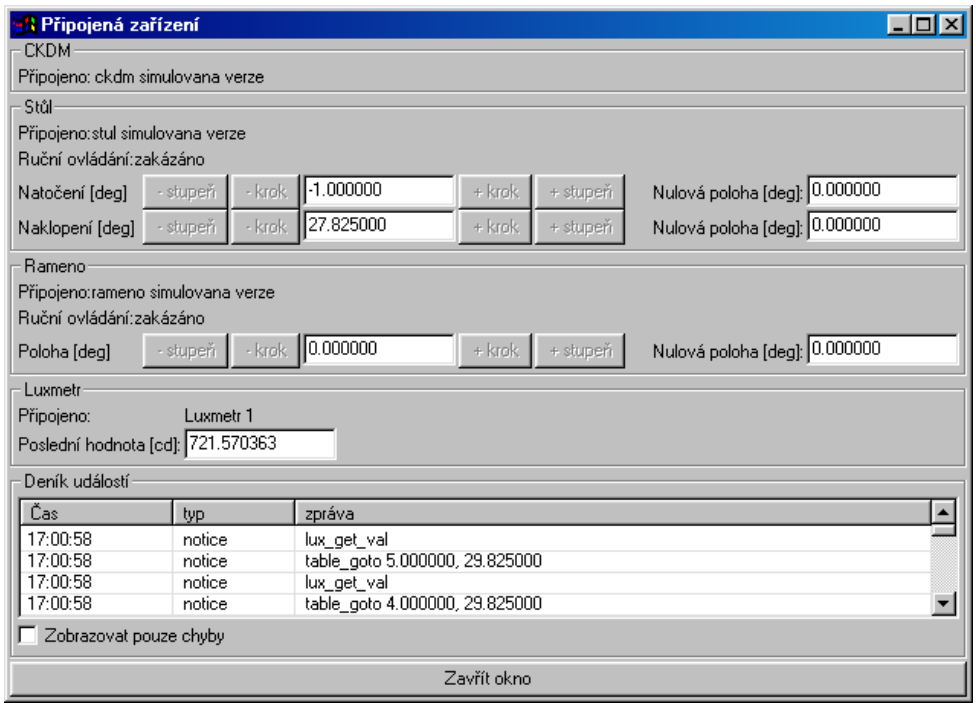
Obrázek 19: Rozšíření měření při vývoji izokandelových návěstidel



Obrázek 20: Výběr tématu luminogramu



Obrázek 21: Příprava na individuální vývoj



Obrázek 22: Stav připojených přístrojů